
Activités
avec l'Amstrad
CPC464
D. Lawrence / S. Lane
MICROPRATIQUES.
AMSTRAD
CPC 464
HACHETTE
Informatique

David Lawrence
Simon Lane

Activités avec l'Amstrad CPC464

Traduit par
Raymond Borraz



Ouvrage original paru sous le titre
THE WORKING AMSTRAD
Publié en langue anglaise par
SUNSHINE BOOKS (an imprint of Scot Press Ltd.)
12/13 Little Newport Street
London, WC2R 3LD

ISBN 2-01-011255-5

HACHETTE *Informatique* 22, rue La Boétie 75008 Paris

Introduction

Cet ouvrage appartient à l'une des collections de micro-informatique ayant remporté le plus de succès. En 1982, lorsque le premier ouvrage « Activités avec le micro » fut publié, peu d'éditeurs pensaient que l'utilisateur courant d'un micro voulait vraiment *comprendre* sa machine, et apprendre à programmer... en programmant. La plupart des livres étaient remplis de jeux et de programmes d'une affligeante banalité, quand ce n'était pas l'inévitable « Vos premiers pas avec... ». L'idée d'un livre présentant un catalogue de programmes sérieux et utiles tout en dévoilant les méthodes utilisées en programmation « sérieuse », n'était pas encore au goût du jour.

Mais, tout comme mon éditeur, j'étais convaincu que les livres « Activités avec le micro » répondaient vraiment à l'aspiration des possesseurs de micros, toujours plus nombreux. Et nous avions raison.

Depuis lors, nos livres ont accompagné les micro-ordinateurs dans pratiquement tous les pays. Leurs traductions, passées ou présentes, couvrent 14 langues différentes.

Et voici qu'apparaît une machine ambitieuse : l'Amstrad CPC 464 avec une nouvelle et brillante version de BASIC, une mémoire plus que suffisante pour pratiquement toutes les applications, et des fonctions sans égales à prix équivalent. Le 464 et le concept des livres « Activités avec le micro » sont parfaitement complémentaires, comme vous allez pouvoir l'apprécier dans les chapitres suivants.

Comment utiliser cet ouvrage

Vous pouvez utiliser ce livre de différentes façons :

1) Comme un répertoire de programmes utiles que vous pouvez adapter et développer pour répondre à vos propres besoins.

2) Comme un répertoire de sous-programmes pouvant servir de fondation à vos propres programmes.

3) Comme une introduction à la programmation avec le BASIC du CPC 464.

Nous tenons à souligner qu'il s'agit *d'un livre* et non d'un simple catalogue de programmes en vrac. Fréquemment, nos lecteurs ont des difficultés parce qu'ils sont passés directement aux programmes plus complexes placés en fin d'ouvrage, sans préparation suffisante. Les premiers programmes du livre, parallèlement à leur utilité et intérêt propre, jouent un rôle pédagogique progressif. Ces premiers programmes sont accompagnés d'explications beaucoup plus détaillées, de sorte qu'en arrivant aux derniers programmes plus complexes, vous aurez bien assimilé les différentes techniques.

QUELQUES PRECISIONS

Commandes concernant la couleur

Tous les programmes du livre ont été écrits et testés sur un Amstrad standard, avec un moniteur couleur. Les lecteurs disposant d'un moniteur noir et blanc devront parfois adapter certaines commandes liées à la couleur.

Mémoire auxiliaire

Le stockage sur mémoire auxiliaire fait appel au lecteur de cassettes incorporé (Datacorder). Si vous faites l'acquisition d'une unité de disques compatible, vous pourrez sans difficulté modifier les commandes d'ouverture et de fermeture en conséquence.

Signe exponentiel

Le symbole ^ qui apparaît dans les programmes est le signe exponentiel.

1

Variations sur les heures

Commencer un livre de ce type n'est jamais simple. Un programme trop complexe, et voilà le lecteur perdu avant d'avoir pu se raccrocher à quelques repères simples qui faciliteront la compréhension des programmes suivants. Par contre, si les premiers programmes sont trop faciles, le lecteur risque de s'arrêter avant d'atteindre les programmes suivants, plus intéressants.

J'ai donc adopté pour ce premier chapitre, un jeu de quatre programmes qui traitent de l'heure, et des différentes façons dont le CPC 464 peut traiter ce problème. Bien que très simples, ces programmes présentent de nombreux concepts que nous utiliserons ultérieurement dans des programmes plus complexes. De plus, par l'usage qu'ils font du calcul, du son, et du graphisme, ces programmes constituent une excellente initiation aux remarquables attributs de base de votre CPC 464.

Dans ce chapitre, nous verrons les programmes suivants :

HEURCLASSIQUE : Une horloge classique en haute résolution.

HEURFANTAISIE : Une autre façon d'indiquer l'heure.

COMPTEUR : 16 compteurs pouvant fonctionner simultanément en déclenchant une alarme et en affichant un message de rappel.

EVENEMENT : Transforme votre CPC 464 en un chronomètre perfectionné (... et plutôt cher).

PROGRAMME 1.1 : HEURCLASSIQUE

Fonction du programme

Ce programme reproduit un cadran d'horloge sur l'écran, avec des aiguilles en mouvement. S'agissant du premier programme, nous vous recommandons de lire avec grande attention les commentaires explicatifs ; vous y apprendrez beaucoup de choses sur les techniques employées dans cet ouvrage.



Note : L'apparence légèrement ovale du cadran est due à la recopie d'écran. Soyez rassuré, le cadran affiché sera parfaitement rond sur votre écran.

Figure 1.1 : Recopie d'écran de Heurclassique

Ce programme va permettre de couvrir les points suivants :

1) Sauvegarde de programmes en cours de développement.
2) Initialisation du programme.
3) Modules de contrôle.
4) Synchronisation avec la commande EVERY.
5) Programmation modulaire.
6) Le module de contrôle.
7) Les notions mathématiques simples liées à un cercle.

Module 1.1.1 : Sauvegarde du programme

Ces trois lignes au début peuvent paraître curieuses, mais ceux qui ont déjà utilisé des livres «Activités» pour des machines précédentes, savent tout l'intérêt de ce petit module, et tous les soucis qu'il peut vous épargner pendant le développement de programmes.

Trop souvent, c'est la cruelle réalité qui nous apprend qu'il **faut** sauvegarder les programmes de façon régulière, au fur et à mesure de leur développement. Tôt ou tard, des heures de travail sont anéanties par une saute de courant, un fusible grillé, un choc donné au micro, ou un pied qui débranche une prise. En pareil cas, l'utilisateur expérimenté ne perdra jamais plus de 15 minutes de travail, tout simplement parce qu'il n'aura pas attendu plus longtemps pour sauvegarder les lignes du programme qu'il vient d'écrire.

Les trois lignes de ce module vous incitent à sauvegarder régulièrement le programme en cours de création en tapant simplement GOTO 2. Un autre petit détail vous fera gagner du temps : un tel module, placé au début de tous vos programmes, donne une ligne de départ standard, c'est-à-dire '1'. Il est souvent bon de commencer un programme avec GOTO, si vous avez défini des variables que vous souhaitez effacer avec RUN. Grâce à ce module, vous n'avez pas à déterminer le numéro de la première ligne, car vous pouvez toujours commencer avec GOTO 1.

Ce module précédait tous les programmes du livre lors de leur développement, mais, comme seul le nom du programme change, nous n'avons pas jugé utile de l'inclure dans les programmes suivants.

MODULE 1.1.1 : LIGNES 1-3

```
1 GOTO 3
2 SAVE «heurclassique» : STOP
3 REM
```

Test

Vérifiez que le lecteur contient une cassette. Entrez

GOTO 2[ENTER]

Mettez le lecteur en route en suivant les indications. Après un bref instant, BREAK IN 2 doit s'afficher. Vous pouvez alors supprimer les trois lignes en mémoire et recharger le programme à partir de la cassette en la rebobinant et en entrant

NEW[ENTER] (efface le programme actuel)
LOAD «HEURCLASSIQUE» [ENTER]

Après le chargement du programme, listez-le. Le module doit se retrouver en place.

Module 1.1.2 : Initialisation du programme

Tout programme digne de ce nom utilise des variables et des constantes, c'est-à-dire des étiquettes (parfois appelées label), dont les valeurs peuvent changer pendant le déroulement du programme ou, à tout le moins, d'un programme à un autre. Grâce aux variables, vous pouvez écrire des lignes de programme adaptées à plusieurs situations. Ainsi, vous pouvez utiliser PRINT 2*A, indépendamment de la valeur de A. Très peu de variables *doivent* absolument avoir une valeur déclarée lors de la première exécution (RUN) du programme. Il est fréquent qu'une valeur de variable ne soit définie qu'en cours de programme, lorsqu'elle est indispensable. En général, il est conseillé de déclarer en tout début de programme la valeur des variables principales, lors de «l'initialisation».

Toutefois, si la mémoire est limitée, il peut être judicieux d'omettre des variables dont la valeur *n'est pas importante* au lancement initial du programme. Dans ce programme, à l'exception des valeurs représentant les couleurs à utiliser, toutes les variables importantes sont fournies par l'utilisateur lorsqu'il est invité à spécifier l'heure. De telles variables, sans intérêt au lancement du programme, sont souvent ignorées (non déclarées) dans le module d'initialisation.

Dans notre cas particulier, le module a pour objet de définir les paramètres d'affichage graphique éventuels et de mettre la machine en mode 'DEGré', qui est adapté aux calculs portant sur des heures.

MODULE 1.1.2 : LIGNES 2000-2110

```
2000 REM ****************
2010 REM *Initialisation*
2020 REM ****************
2030 MODE 1
```

```
2040 INK 0,1
2050 INK 1,24
2060 INK 2,3
2070 INK 3,6
2080 BORDER 1
2090 ORIGIN 200,200
2100 DEG
2110 RETURN
```

Test

Pour un premier test non approfondi du module, vous pouvez entrer :

GOTO 2000

Si le module a été entré correctement, l'écran s'efface et le message suivant apparaît : 'Unexpected RETURN in 2110' (RETURN inattendu en 2110). Vous ne pourrez tester de façon approfondie le fonctionnement du module qu'après avoir entré les modules suivants.

Module 1.1.3 : Introduction de l'heure

Avant de créer une horloge, nous devons bien entendu donner l'heure. Ce module permet à l'utilisateur de donner l'heure et les minutes, et de les stocker dans la mémoire du 464 sous la forme de deux variables : HEURE et MINUTE.

MODULE 1.1.3 : LIGNES 3000-3080

```
3000 REM ***************************
3010 REM *Introduction de l'heure*
3020 REM ***************************
3030 PRINT "Mise à l'heure :":PRINT
3040 INPUT "Heure (1-12) :";heure
3050 IF heure<1 OR heure>12 THEN PRINT "
** Hors limites : essayer à nouveau **":
GOTO 3040
3060 INPUT "Minute (0-59) :";minute
3070 IF minute<0 OR minute>59 THEN PRINT
"**Hors limites : essayer à nouveau **":
GOTO 3060
3080 RETURN
```

Test

Entrez

GOTO 3000[ENTER]

Le système vous demande l'heure, en heures et minutes. Les valeurs fournies doivent être correctes, sous peine d'obtenir un message d'erreur. Dès qu'une heure est acceptée, le programme s'arrête et affiche un message d'erreur 'Unexpected RETURN' (RETURN inattendu). Rien d'anormal à cela : après l'entrée du module final, ce module deviendra un sous-programme, appelé par un GOSUB.

Pour pousser plus loin la vérification, vous pouvez taper :

PRINT heure,minute[ENTER]

qui doit afficher les valeurs des heures et minutes que vous avez entrées.

Module 1.1.4 : Création du cadran de l'horloge

Après avoir entré l'heure, nous allons tracer le cadran de l'horloge, sur lequel les modules suivants placeront les aiguilles.

MODULE 1.1.4 : LIGNES 4000-4090

```
4000 REM *********************
4010 REM *Cadran de l'horloge*
4020 REM *********************
4030 CLS
4040 FOR a=0 TO 359 STEP 6
4050 MOVE 200*SIN(a),200*COS(a)
4060 r=195:IF a MOD 30=0 THEN r=185
4070 DRAW r*SIN(a),r*COS(a),1
4080 NEXT a
4090 RETURN
```

Commentaire

Définition d'un cercle

La méthode employée dans ce module repose sur le fait qu'il est possible de déterminer tout point sur la circonférence d'un cercle, à partir des données suivantes :

a) Le rayon du cercle. (RADIUS)
b) L'angle à parcourir vers la droite à partir de la position trois heures, pour atteindre le point spécifié. (ANGLE)
c) Les coordonnées du centre du cercle. (CENTRE X et CENTRE Y)

A partir de ces trois éléments, deux formules permettent d'exprimer la position :

coordonnée X = RADIUS*COSINE(ANGLE)+CENTRE X
coordonnée Y = RADIUS*SINE(ANGLE)+CENTRE Y

Il n'est pas dans notre propos d'approfondir ce domaine, mais tout bon manuel d'introduction à la trigonométrie vous donnera tous les détails à ce sujet. Dans notre cas particulier, les formules sont même plus simples, car, dans le module d'initialisation, nous avons déplacé l'origine (ORIGIN) graphique de l'écran sur la position 200,200. Autrement dit, toute référence à la position 0,0 sur l'écran apparaîtra, à partir de l'angle inférieur gauche de l'écran, sur la position 200 vers le haut et 200 vers la droite (position d'un pixel) (1). Donc, comme nous avons déplacé l'origine graphique sur la position correspondant au centre du cadran, il n'est plus utile de faire référence aux coordonnées X et Y du centre du cadran. Toutes deux sont égales à zéro. Les formules donnant la position d'un point donné sur la circonférence du cercle deviennent à présent :

coordonnée X = RADIUS*COSINE(ANGLE)
coordonnée Y = RADIUS*SINE(ANGLE)

C'est exactement cela que nous utiliserons dans ce module.

Lignes 4040-4080 : Cette boucle nous fait parcourir les 360 degrés d'un cercle, par sauts de six degrés. Comme il y a 60 minutes dans une heure et que 360/60 = 6, vous devinez que la boucle a un lien direct avec la représentation des minutes sur le cadran, tracées avec les lignes suivantes.

Ligne 4050 : Le curseur graphique est déplacé (MOVE) sur un point de la circonférence du cercle, en utilisant les formules précédentes. La première position, la variable A de la boucle étant à zéro, sera au sommet du cercle. Les positions suivantes progresseront sur le cercle vers la droite, par incréments de six degrés.

Lignes 4060-4070 : Ensuite, une ligne est tracée à partir des positions de la circonférence du cercle, vers le centre. En fait,

---

(1) pixel : contraction de «picture element» = élément de dessin.

les lignes sont tracées entre les circonférences de deux cercles de tailles différentes. Le cercle extérieur a un rayon de 200 pixels (points). La taille du cercle intérieur varie selon que l'angle indiqué par la variable A de la boucle, pointe sur cinq minutes pleines (30 degrés) ou non. Si l'angle ne pointe pas sur cinq minutes pleines, le cercle intérieur n'est que de cinq pixels à l'intérieur du cercle d'origine : la marque tracée n'aura que cinq pixels de long.

Grâce à la fonction MOD, le programme reconnaît lorsque un angle de 30 degrés. Par exemple, 5 MOD 2 donne le résultat 1, soit le reste de la division de 5 par 2. Un MOD 30 comme dans le programme, donne le reste obtenu lorsque la variable A de la boucle est divisée par 30 ; si le reste est zéro, l'angle est exactement divisible par 30. Dans ce cas, le cercle intérieur est plus petit ou, si vous préférez, la marque tracée est plus longue, ce qui met en évidence les graduations de cinq minutes.

Test

Entrez

GOTO 4000[ENTER]

L'écran doit s'effacer, puis afficher un cadran d'horloge. Le programme se termine par un message d'erreur 'Unexpected RETURN' (RETURN inattendu).

Module 1.1.5 : Réglage des minutes et des heures

Ce module nous amène à l'essentiel du programme : le calcul de l'heure actuelle et, à partir de là, des coordonnées correspondant aux aiguilles de l'horloge. Toutefois, pour une utilisation correcte, ce module doit attendre que le dernier module ait été entré. Ce n'est qu'alors qu'il sera appelé à intervalles corrects, c'est-à-dire une fois par minute.

MODULE 1.1.5 : LIGNES 5000-5130

```
5000 REM ********************
5010 REM *Réglage de l'heure*
5020 REM ********************
5030 c=0
5040 angle=minute*6:taille=180:GOSUB 600
0
5050 angle=heure*30+minute/2:taille=120:
```

```
GOSUB 6000
5060 minute=minute+1
5070 IF minute=60 THEN minute=0:heure=he
ure+1
5080 IF heure=13 THEN heure=1
5090 c=2
5100 angle=minute*6:taille=180:GOSUB 600
0
5110 c=3
5120 angle=heure*30+minute/2:taille=120:
GOSUB 6000
5130 RETURN
```

Commentaire

Ce module est en fait divisé en trois sections distinctes. La première section a pour tâche d'appeler un module suivant qui efface les aiguilles sur leur position déjà existante. La deuxième section ajoute 1 à MINUTE et modifie HEURE en conséquence. La troisième section prend les nouvelles valeurs heure/minute et appelle un module suivant qui retrace les aiguilles.

Lignes 5030-5050 : Le module suivant utilisera la variable C pour définir la couleur de l'encre lorsque les aiguilles seront tracées. Elle est d'abord définie avec la valeur 0. En revenant au module d'initialisation, vous observerez que la couleur 0, la couleur de fond, a été définie en bleu. La ligne 5040 définit les paramètres de l'aiguille des minutes, soit son angle et sa taille. La valeur des minutes n'ayant pas changé, le fait de tracer l'aiguille dans la couleur de fond aura pour résultat d'effacer l'aiguille existante. La ligne 5050 fait de même pour l'aiguille des heures.

Lignes 5060-5080 : Les valeurs des minutes et des heures sont mises à jour. Pour cela, 1 est ajouté aux minutes et certains ajustements permettent de s'assurer que les valeurs résultantes sont réalistes.

Lignes 5090-5120 : Les paramètres des aiguilles des minutes et des heures sont redéfinis avec les valeurs mises à jour, avant que le module suivant soit appelé pour les retracer. Le numéro de couleur C prend la valeur 2 pour l'aiguille des minutes, et 3 pour l'aiguille des heures. En revenant au module d'initialisation, vous pouvez observer que ces valeurs correspondent respectivement au rouge et au rouge vif.

Test

Tapez :

6000 RETURN[ENTER]

Il s'agit d'une ligne temporaire qui tient compte du fait que le module que vous venez d'entrer en appelle un qui n'existe pas encore.

A présent, tapez :

HEURE=0[ENTER]
MINUTE=50[ENTER]
GOTO 5000[ENTER]

L'exécution doit s'arrêter presque instantanément, avec un message d'erreur 'Unexpected RETURN' (RETOUR inattendu). Tapez ensuite :

PRINT minute,heure

Vous devriez obtenir :

0 1
indiquant que vos 59 minutes d'origine sont passées à 60 et que la valeur de l'heure a augmenté en conséquence.

Module 1.1.6 : Tracé des aiguilles

Après avoir fait les calculs nécessaires pour obtenir la position des aiguilles, nous pouvons passer à leur traçage proprement dit.

Module 1.1.6 : Lignes 6000-6110

```
6000 REM ********************
6010 REM *Tracé des aiguilles*
6020 REM ********************
6030 taille2=taille*2/3
6040 PLOT 0,0,c
6050 DRAW taille2*SIN(angle-8),taille2*C
OS(angle-8)
6060 DRAW taille*SIN(angle),taille*COS(a
ngle)
6070 DRAW taille2*SIN(angle+8),taille2*C
OS(angle+8)
6080 DRAW 0,0
```

```
6090 MOVE taille2*SIN(angle),taille2*COS
(angle)
6100 GOSUB 7000
6110 RETURN
```

Commentaires

Ligne 6030 : Les aiguilles se présentent sous la forme d'un losange quelque peu étiré. SIZE2 représente la distance à partir du centre, de l'endroit où l'aiguille commence à se rétrécir vers le point.

Ligne 6040 : Le curseur graphique est déplacé au centre du cadran et la couleur C est définie pour le tracé, de sorte que le module précédent commande la couleur de tout tracé à venir.

Lignes 6050-6080 : Ces lignes tracent sur l'écran quatre lignes qui définissent l'aiguille. La première ligne a une longueur de SIZE2 et est tracée avec un angle de huit degrés vers la gauche, vers l'angle correct pour les heures ou les minutes. La ligne suivante est tracée de la fin de la première ligne vers une position située à l'angle correct et à SIZE pixels du centre. La troisième ligne est tracée vers une position de SIZE2 pixels à partir du centre, et à huit degrés vers la droite en direction de l'angle correct. Pour finir, une ligne tracée vers le centre termine le dessin.

Lignes 6090-6100 : Ces deux lignes sont nécessaires au module suivant, qui colorie les formes vides tracées par ce module. La ligne active est la ligne 6090 : elle déplace le curseur graphique sur un point clairement situé à l'intérieur de l'aiguille qui a été tracée.

Test

D'abord, il faut ajouter une ligne temporaire représentant le module suivant :

7000 RETURN

Puis tapez :

CLEAR :CLS :DEG :GOTO 5000[ENTER]

L'écran s'efface et les deux aiguilles de l'horloge représentent à peu près douze heures et une minute. Les aiguilles sont simplement tracées; c'est le module suivant qui les remplira de couleurs.

Module 1.1.7 : Remplissage avec des couleurs

Par rapport à d'autres ordinateurs personnels, le 464 présente une lacune : l'absence d'une commande permettant de remplir une forme donnée avec une couleur. Le module qui suit est bien plus lent qu'une commande intégrée et présente quelques limites ; néanmoins, il donne le résultat souhaité.

La routine de ce module peut remplir tout polygone dont les angles sont concaves, vus de l'extérieur ; autrement dit, les angles pointent vers l'extérieur et non vers l'intérieur. Elle peut remplir *certaines* formes ayant des angles pointant vers l'intérieur, mais pas toutes. Pour utiliser cette routine sur des formes aussi irrégulières, vous devrez remplir successivement plusieurs sections. En particulier, si vous utilisez cette routine sur une forme non fermée, elle recherchera les bords de la figure à l'extérieur de l'écran.

MODULE 1.1.7 : LIGNES 7000-7300

```
7000 REM *************
7010 REM *Remplissage*
7020 REM *************
7030 IF TESTR(0,0)=c THEN RETURN
7040 s=2
7050 MOVE 2*INT(XPOS/2),2*INT(YPOS/2)
7060 :
7070 :
7075 REM ***********************
7080 REM *Recherche haut/droite*
7085 REM ***********************
7090 IF TESTR(0,s)<>c THEN GOTO 7090
7100 IF TESTR(s,-s)<>c THEN GOTO 7090
7110 :
7120 :
7125 REM ***********************
7130 REM *Recherche haut/gauche*
7135 REM ***********************
7140 MOVE -s,0
7150 IF TESTR(0,s)<>c THEN GOTO 7090
7160 IF TESTR(-s,-s)<>c THEN GOTO 7150
7170 :
7180 :
7185 REM **********************
7190 REM *Coloriage des lignes*
7195 REM **********************
```

```
7200 x1=XPOS
7210 MOVER s,0
7220 PLOTR 0;0,c
7230 IF TESTR(s,0)<>c THEN GOTO 7220
7240 x2=XPOS-s
7250 MOVE x1,YPOS-s
7260 IF TESTR(s,0)<>c THEN GOTO 7290
7270 IF XPOS=x2 THEN RETURN
7280 GOTO 7260
7290 IF TESTR(-s,0)=c THEN GOTO 7200
7300 GOTO 7290
```

Commentaires

Ligne 7030 : Comme la plupart des autres routines de «coloriage» ou de «remplissage», cette routine ne fonctionnera pas si la couleur du pixel où elle doit démarrer est déjà attribuée à la routine pour le remplissage.

Ligne 7040 : La variable S représente le déplacement horizontal minimum utilisé en cours de programme. Sa valeur peut être modifiée si d'autres modes graphiques sont utilisés.

Ligne 7050 : En mode graphique 1, utilisé pour ce programme, les pixels sont groupés par paires. Cette ligne garantit que, quel que soit le sort des coordonnées de tracé, elles représentent des nombres pairs.

Lignes 7080-7100 : Le programme recherche désormais une partie de la bordure de la figure. Au premier abord, ces lignes peuvent surprendre, car aucun mouvement n'est apparent, et il semble n'y avoir qu'une ligne dans une boucle sans fin. En fait, TEST et TESTR(elative) déplacent le curseur graphique sur la position spécifiée. Donc la première ligne recherche vers le haut, en ligne droite, le premier pixel de couleur correcte. Dès qu'une partie de la bordure a été trouvée, la ligne suivante fait un pas vers le bas et vers la droite, pour voir si ce point est vacant. S'il *est* vacant, la première ligne est rappelée pour voir s'il est possible d'aller plus haut.

Lignes 7130-7160 : Après avoir recherché au maximum vers le haut et vers la droite, ces lignes commencent à explorer vers la gauche, recherchant une façon de poursuivre vers le haut. A la fin de ces lignes et du groupe précédent, le programme a atteint soit le sommet de la figure, soit une impasse parce que la figure n'est pas régulière.

Lignes 7200-7230 : Lorsque le point le plus haut a été trouvé, le «remplissage» peut vraiment commencer. Les commandes MOVER(elative) et PLOTR(elative) permettent de tracer une ligne de pixels de gauche à droite jusqu'à ce qu'une autre partie de la bordure soit atteinte.

Lignes 7240-7250 : Au terme d'une ligne horizontale, la variable X2 reçoit une valeur égale à la position la plus à droite remplie, et le curseur revient au début de la ligne, mais un pixel plus bas.

Lignes 7260-7280 : Ce test a lieu lorsque le curseur est ramené vers la gauche. Si le pixel ainsi atteint présente déjà la couleur de remplissage, le programme explore de gauche à droite à la recherche du premier pixel vierge. Si la recherche s'étend audelà de la fin de la ligne précédente, qui vient d'être remplie, le bas de la figure a été atteint.

Lignes 7290-7300 : Si, pendant le déplacement du curseur vers le bas et vers la gauche pour commencer une nouvelle ligne, le pixel atteint est vierge, le programme cherche vers la gauche pour atteindre la bordure de la figure, avant de commencer le remplissage.

Test

Pour tester ce module, la méthode la plus simple consiste à suivre la procédure de test du module précédent. Toutefois, le résultat sera différent; en effet, les deux aiguilles sont à présent remplies avec la couleur correcte.

Module 1.1.8 : Regroupement

A ce stade, peut-être vous posez-vous quelques questions sur la méthode d'écriture du programme. Pourquoi ne pas avoir rassemblé toutes les fonctions que nous avons décrites et les avoir mises en oeuvre avec quelques instructions GOTO? Hélas, de nombreux programmes du commerce sont construits de la sorte.

Dans ce livre, vous constaterez que tous les programmes sont constitués de modules clairement identifiables. La raison en est que ces programmes sont plus faciles à lire, à mettre au point et à modifier en remplaçant des modules si vous apprenez de nouvelles méthodes, et qu'ils peuvent recevoir de nouveaux modules. Ces programmes peuvent vous apprendre beaucoup, mais la leçon la plus importante sera la technique de programmation modulaire.

Le module actuel est la clé de voûte de l'ensemble car, lorsque tous les autres modules ont été entrés et testés, il en faut un pour contrôler le tout. En fait, le module que vous êtes sur le point d'entrer peut être considéré comme le programme proprement dit, tout le reste n'en étant qu'une extension.

MODULE 1.1.8 : LIGNES 1000-1100

```
1000 REM **********
1010 REM *Controle*
1020 REM **********
1030 GOSUB 2000
1040 GOSUB 3000
1050 GOSUB 4000
1060 GOSUB 5000
1070 EVERY 3000 GOSUB 5000
1080 IF INKEY$<>" " THEN 1080
1090 CLS
1100 END
```

Commentaires

Ligne 1070 : La clé du programme. Cette seule ligne donne la synchronisation du programme, en utilisant la puissante commande EVERY du 464. Cette commande permet à l'utilisateur de spécifier une durée, puis, lorsque cette durée est écoulée, d'interrompre toute tâche en cours d'exécution pour passer à autre chose. Lorsque l'interruption est terminée, la tâche originale, quelle qu'elle soit, recommence jusqu'à l'interruption suivante. La période après laquelle l'interruption a lieu est découpée en 50èmes de seconde ; cette ligne demande donc au 464 d'appeler le module de la ligne 5000, une fois par minute, ce qui augmente la valeur des minutes et retrace les aiguilles.

Lignes 1080-1100 : Le reste du temps, le programme est prisonnier de la boucle sans fin représentée par la ligne 1080, attendant que l'utilisateur actionne la barre d'espacement. Alors, l'écran s'efface et le programme se termine.

Test

A présent, le programme doit être entièrement opérationnel. Lancez-le, entrez l'heure, et le cadran doit s'afficher avec les aiguilles dans la bonne position. Vous pouvez vérifier plus

rapidement le bon fonctionnement de l'affichage en supprimant deux zéros de la valeur 3000 à la ligne 1070. Les aiguilles seront alors tracées et retracées en permanence.

PROGRAMME 1-2 : HEURFANTAISIE

Fonction du programme

L'un des aspects les plus séduisants des ordinateurs avec affichage graphique comme le CPC 464 est qu'ils vous permettent d'afficher, en toute fantaisie, des informations de façon nouvelle en laissant libre cours à votre imagination. A partir d'une horloge standard, le programme suivant donne une vue très différente de l'heure. Ici, nous allons représenter les heures et les minutes par deux lignes qui s'étendent de gauche à droite et de haut en bas, divisant l'écran en quatre rectangles de couleurs différentes. Une bonne partie de Heurfantaisie est identique à Heurclassique, aussi les explications seront-elles réduites.

Ce programme couvre les nouveaux points suivants :

1) Fenêtres
2) Formatage des nombres
3) Découpage de chaînes

Module 1.2.1 : Initialisation

Ce simple module d'initialisation permet de définir les couleurs nécessaires à l'affichage. Le rôle de la fenêtre définie à la ligne 2100 sera expliqué dans les commentaires d'un module ultérieur.

MODULE 1.2.1 : LIGNES 2000-2120

```
2000 REM ****************
2010 REM *Initialisation*
2020 REM ****************
2030 MODE 1
2040 INK 0,1
2050 INK 1,24
2060 INK 2,9
2070 INK 3,6
```

```
2080 BORDER 1
2090 PAPER 0:PEN 1
2100 WINDOW #1,39,39,9,18
2110 PAPER #1,0:PEN #1,1
2120 RETURN
```

Test

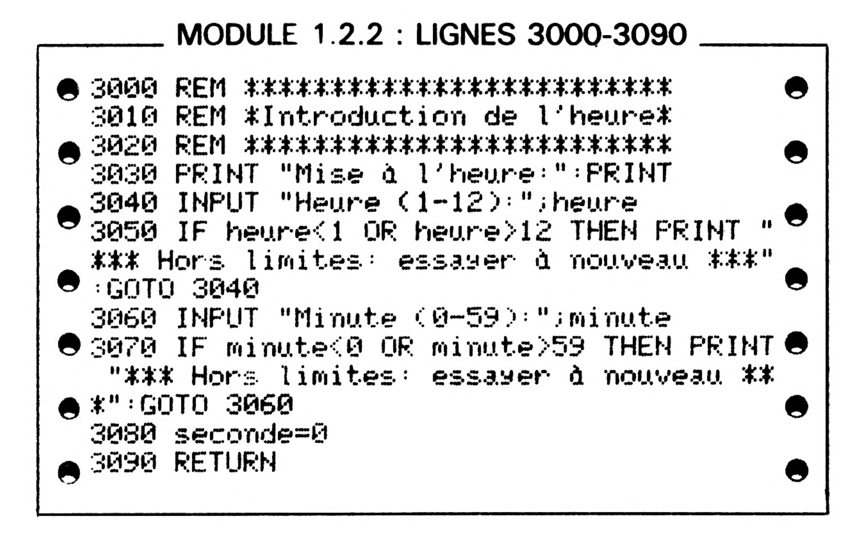
Comme avec le premier programme, le seul test possible à ce stade consiste à taper :

GOTO 2000[ENTER]

Le programme doit se terminer par un message d'erreur 'Unexpected RETURN' (RETURN inattendu). Tout autre type d'erreur signalera une erreur pendant l'entrée du module.

Module 1.2.2 : Entrée de l'heure

Module identique à celui de Heurclassique.



MODULE 1.2.2 : LIGNES 3000-3090

```
3000 REM ************************
3010 REM *Introduction de l'heure*
3020 REM ************************
3030 PRINT "Mise à l'heure:":PRINT
3040 INPUT "Heure (1-12):";heure
3050 IF heure<1 OR heure>12 THEN PRINT "
*** Hors limites: essayer à nouveau ***"
:GOTO 3040
3060 INPUT "Minute (0-59):";minute
3070 IF minute<0 OR minute>59 THEN PRINT
 "*** Hors limites: essayer à nouveau **
*":GOTO 3060
3080 seconde=0
3090 RETURN
```

Module 1.2.3 : Définition de la bordure de l'écran

Ce module affiche une grille d'heures et de minutes sur la bordure supérieure gauche de l'écran.

MODULE 1.2.3 : LIGNES 4000-4120

```
4000 REM ************************
4010 REM *Mise en Place affichage*
4020 REM ************************
4030 CLS:PRINT" ";
4040 FOR i=5 TO 55 STEP 10
4050 PAPER 0:PEN 1:PRINT USING"###";i;
4060 PAPER 1:PEN 0:PRINT USING"###";i+5;
4070 NEXT i
4080 PAPER 0:PEN 1
4090 FOR i=1 TO 12
4100 LOCATE 1,i*2+1:PRINT USING"##";i
4110 NEXT i
4120 RETURN
```

Commentaires

Lignes 4040-4070 : La boucle affiche les valeurs des minutes sur la bordure supérieure de l'écran, en faisant alterner les lettres normales et inverses (couleurs du papier et de la plume permutées). PRINT USING permet d'assurer que chaque chiffre de cinq minutes est affiché avec trois caractères; comme tous les nombres ont moins de trois caractères de long, PRINT USING les complète avec un ou deux espaces à gauche.

Lignes 4080-4110 : La variable I de la boucle, conjointement à la commande LOCATE, permet d'afficher les heures sur la bordure gauche de l'écran.

Test

Pour bien tester le module, vous devez insérer une ligne temporaire :

4115 GOTO 4115

Cela empêche l'écran de défiler vers le haut avec un message 'Unexpected RETURN' (RETURN inattendu).

Ensuite entrez :

GOTO 4000[ENTER]

et la grille s'affiche à la périphérie de l'écran. Lorsque l'affichage vous convient, appuyez deux fois sur ESCAPE. Et n'oubliez pas de supprimer la ligne temporaire.

Module 1.2.4 : Création d'une chaîne d'heure

L'une des fonctions du programme consiste à afficher l'heure de façon numérique (digitale) directe, ainsi que par la méthode plus originale décrite précédemment. La chaîne est basée sur des variables créées par le module suivant, mais vous devez entrer celui-ci en premier lieu, pour permettre l'exécution correcte du modèle suivant.

MODULE 1.2.4 : LIGNES 7000-7050

```
7000 REM ************************
7010 REM *Creation chaine d'heure*
7020 REM ************************
7030 tim$=STR$(1000000+heure*10000+minut
e*100+seconde)
7040 tim$=MID$(tim$,3,2)+" "+MID$(tim$,5
,2)+" "+MID$(tim$,7,2)
7050 RETURN
```

Commentaires

Lignes 7030-7040 : Une méthode astucieuse pour combiner une suite de nombres en un seul nombre, consiste à multiplier chaque nombre par des puissances décroissantes de 10 puis de les additionner avec une puissance de 10 supérieure. Si par exemple HEURE = 1, MINUTE = 36 et SECONDE = 2, le résultat de cette ligne serait 1013602. Pourquoi ? Observez le chiffre des heures : le 1000000 qui a été ajouté a placé un 0 devant le seul '1' de la valeur de l'heure, et devant le '2' des secondes.

Nous utilisons ensuite un 'découpage de chaîne' pour découper le nombre en trois sections de deux chiffres, sachant qu'un 0 à gauche a été ajouté si l'une des valeurs d'origine ne comptait qu'un chiffre. Les fonctions de chaîne de la deuxième ligne extraient les trois figures en spécifiant la position de départ dans le nombre créé, et le nombre de caractères à extraire. Ainsi, MID$(A$,3,2) signifie la partie de A$ qui comprend deux caractères commençant au caractère 3.

Si vous examinez les commandes de découpage de chaîne de la ligne 7040, vous serez surpris du fait que la valeur des heures est prise à partir du caractère 3, alors qu'elle part certainement du chiffre 2 du nombre artificiel créé à la ligne 7030.

Cette contradiction apparente s'explique par le fait que, pour extraire les valeurs sous la forme d'une chaîne, nous avons d'abord transformé le nombre en une chaîne avec la fonction STR$. Dans ce cas, le nombre a le même aspect, mais il peut être traité en tant que chaîne; la seule différence étant que, s'il s'agit d'un nombre positif, un espace est ajouté pour compenser l'absence d'un signe '+'. Le premier caractère utilisable d'un nombre converti en une chaîne grâce à STR$ est le caractère numéro 2.

Cette manipulation se traduit par une courte chaîne sous la forme :

DEUX CHIFFRES D'HEURES/ESPACE/DEUX CHIFFRES DE MINUTES/ESPACE/
DEUX CHIFFRES DE SECONDES

Test

Tapez :
HEURE = 1[ENTER]
MINUTE = 1[ENTER]
SECONDE = 1[ENTER]
GOTO 7000[ENTER]

Lorsque le programme affiche une erreur 'Unexpected RETURN' (RETURN inattendu), tapez :

PRINT tim$[ENTER]

vous devez obtenir :

01 01 01

Module 1.2.5 :
Réglage de l'heure

Ce module est parallèle à son homologue du programme précédent.

MODULE 1.2.5 : LIGNES 6000-6160

```
6000 REM ********************
6010 REM *Réglage de l'heure*
6020 REM ********************
6030 seconde=seconde+1
6040 WHILE seconde>59
```

```
6050 seconde=0:minute=minute+1
6060 WHILE minute>59
6070 minute=0:heure=heure+1
6080 WHILE heure>12
6090 heure=1
6100 WEND
6110 WEND
6120 WEND
6130 LOCATE #1,1,1
6140 GOSUB 7000
6150 PRINT#1,tim$;
6160 RETURN
```

Commentaires

Lignes 6030-6120 : Ces trois boucles imbriquées traversent clairement la suite des étapes nécessaires lorsque la deuxième valeur est incrémentée de 1. Chaque boucle successive n'est activée que si, après le changement des secondes, la valeur des minutes ou des heures doit être ajustée, c'est-à-dire si la fin de la minute ou de l'heure a été dépassée.

Lignes 6130-6160 : Comme nous avons créé une chaîne représentant l'heure au module précédent, ces lignes s'affichent verticalement. Grâce à la fenêtre définie dans le module d'initialisation, cette opération est fort simple. Si vous revenez au module d'initialisation, vous constatez que la fenêtre définie se trouve à droite de l'écran et qu'elle n'a qu'un caractère de large. Il s'ensuit que tout ce qui s'affiche dans cette fenêtre, est une succession de caractères les uns sous les autres.

Test

Tapez

CLEAR
SECONDE=59
MINUTE=59
HEURE=12
GOTO 6000[ENTER]

et vous devez voir à nouveau un message 'Unexpected RETURN' (RETURN inattendu) avec '01 00 00' affiché verticalement sur le bord droit de l'écran. Cela démontre bien que les boucles de lá première partie du programme mettent correctement à jour les trois valeurs.

Module 1.2.6 : Affichage de l'heure

Dans ce module, nous en arrivons à l'affichage des rectangles qui vont représenter l'heure. Ce module se propose de présenter une ligne qui balaie l'écran horizontalement pour représenter les minutes, et une autre qui descend pour représenter les heures. Pour cela, nous avons divisé l'écran en quatre sections rectangulaires, deux rouges, deux vertes, les bords de ces rectangles représentant les lignes des heures et des minutes, comme à la figure 1.2.

Bien sûr, nous pourrions placer chacun de ces rectangles directement sur l'écran à l'endroit correct, mais ce serait une difficulté inutile. Il nous suffit de tracer sur la partie supérieure de l'écran des lignes d'espaces de couleurs qui passent du rouge au vert à l'endroit approprié, et sur la partie inférieure de l'écran des lignes qui passent du vert au rouge. Les lignes elles-mêmes seront créées par une simple boucle et par la commande LOCATE.

RECTANGLE 1 (ROUGE)	L I G N E D E S	RECTANGLE 2 (VERT)
LIGNES DES HEURES RECTANGLE 3 (VERT)	M I N U T E S	RECTANGLE 4 (ROUGE)

Figure 1.2 : L'écran est divisé en quatre sections rectangulaires

MODULE 1.2.6 : LIGNES 5000-5100

```
5000 REM **********************
5010 REM *Affichage de l'heure*
5020 REM **********************
```

```
5030 PAPER 2:PEN 3
5040 FOR i=2 TO 25
5050 IF i=INT(heure*2+minute/30+2)THEN P
APER 3:PEN 2
5060 LOCATE 3,i
5070 PRINT STRING$(minute*0.6,CHR$(143))
;TAB(39)
5080 NEXT i
5090 RETURN
5100 RETURN
```

Commentaires

Ligne 5030 : Les couleurs verte et rouge sont respectivement choisies pour la couleur du papier et pour la couleur d'impression.

Lignes 5040-5080 : 24 lignes vont s'afficher, chaque ligne représentant une demi-heure, à partir de la ligne 2 de l'écran (numérotée à partir de 0).

Ligne 5050 : Si la valeur de la boucle a atteint la position où devrait se trouver la ligne des heures, les couleurs du papier et de l'impression sont inversées.

Lignes 5060-5070 : Ces deux lignes localisent et affichent une des lignes sur l'écran. La fonction STRING$ permet de créer une ligne d'espaces inversés (CHR$(143)) jusqu'à la ligne de séparation des minutes. Le TAB en fin de ligne déplace la position d'affichage à la fin de la ligne, en donnant à tous les espaces inutilisés la couleur PAPER en cours. La longueur de chaîne créée sera basée sur la valeur de la variable MINUTE, mais sera ajustée (multipliée par 0.6) pour tenir compte du fait que l'affichage a 36 caractères de large, et non pas 60.

Test

Entrez, en mode direct :

HEURE=6[ENTER]
MINUTE=30[ENTER]
GOTO 5000

L'écran doit se diviser en quatre rectangles à peu près égaux, d'après les lignes de l'illustration précédente. Le programme s'arrête alors avec un message d'erreur 'Unexpected RETURN' (RETURN inattendu).

Module 1.2.7 : Regroupement

Après avoir entré tous les éléments consécutifs du programme, il ne nous reste plus qu'à créer un module de contrôle qui les exécute dans l'ordre correct.

MODULE 1.2.7 : LIGNES 1000-1090

```
1000 REM *********
1010 REM *Controle*
1020 REM *********
1030 ON BREAK GOSUB 1090
1040 GOSUB 2000
1050 GOSUB 3000
1060 EVERY 50 GOSUB 6000
1070 GOSUB 4000
1080 GOSUB 5000:GOTO 1080
1090 PEN 1:PAPER 0:CLS:CLEAR:END
```

Commentaires

Ligne 1030 : La commande ON BREAK GOSUB nous permet de définir l'action du programme si nous appuyons deux fois sur la touche ESCAPE. Dans ce cas, nous choisissons de sauter à la dernière instruction du programme. Les couleurs sont réinitialisées, la mémoire effacée et le programme s'arrête.

Ligne 1060 : Nous retrouvons ici la commande EVERY qui synchronise le programme. A chaque seconde, le module de réglage de l'heure est appelé.

Test

Vous pouvez à présent exécuter la totalité du programme, introduire l'heure et la voir s'afficher.

PROGRAMME 1.3 : COMPTEUR

Fonction du programme

Ce programme vous fournit 16 compteurs «à rebours» dont chacun peut être programmé séparément pour émettre une sonnerie après une période spécifiée *et* indiquer par un bref message la raison de la sonnerie.

Vous pourrez constater que la présentation de ce programme est quelque peu différente des précédents. En effet, celui-ci étant plus long et plus complexe, nous l'avons complété par des lignes de programme vierges permettant d'identifier ses diverses unités (ou segments). Ces lignes vierges ne sont pas indispensables au fonctionnement du programme, mais elles contribuent à sa clarté. Ce programme et quelques autres sont les seuls présentés de cette façon, tout simplement parce que cette technique appliquée à tous les programmes aurait allongé inutilement l'ouvrage. Toutefois, efforcez-vous de l'appliquer dans tous vos programmes.

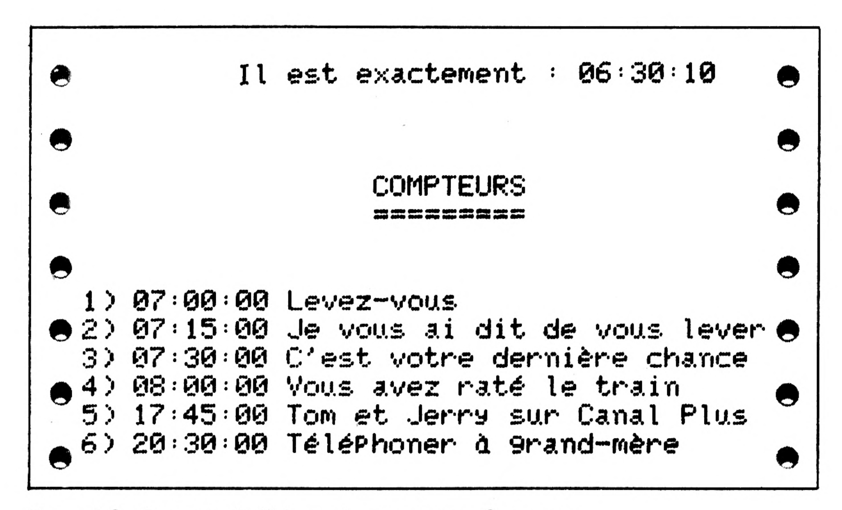


Figure 1.3 : Exemple d'affichage du programme Compteur.

Nouvelles techniques présentées dans ce programme :

1) Le module 'menu'
2) L'utilisation simple de SOUND
3) Insertion et suppression dans des tableaux
4) Création d'un état d'attente

Module 1.3.1 : Initialisation

Ce module d'initialisation est plus complexe que celui des deux programmes précédents, en raison de la plus grande difficulté du programme lui-même. Ce module est surtout consacré à la préparation de l'écran, y compris deux fenêtres, à l'usage du reste du programme.

MODULE 1.3.1 : LIGNES 2000-2150

```
2000 REM ***************
2010 REM *Initialisation*
2020 REM ***************
2030 MODE 1
2040 GOSUB 12000
2050 WINDOW 1,40,25,25
2060 PAPER 0:PEN 1:CLS
2070 WINDOW #1,12,29,1,1
2080 PAPER #1,0:PEN #1,1:CLS #1
2090 PRINT #1,"Il est exactement:"
2100 WINDOW #2,1,40,3,23
2110 PAPER #2,3:PEN #2,2:CLS #2:PRINT #2
2120 SPEED INK 50,25
2130 DIM theure(15),tminute(15),thoraire
(15),message$(15)
2140 thoraire(0)=-1
2150 RETURN
```

Commentaires

Ligne 2040 : Contrairement aux programmes précédents, ce module ne contient pas d'instructions INK. La raison en est que les couleurs INK seront, à l'occasion, modifiées en cours de programme. Cette instruction GOSUB appelle la petite routine qui définit les couleurs de départ : nous entrerons cette sous-routine immédiatement après l'actuelle.

Ligne 2050 : La commande WINDOW peut être utilisée de deux façons. Nous pouvons réserver une zone de l'écran qui sera appelée par un 'numéro de flux' - méthode utilisée dans le programme Heurfantaisie - ou nous pouvons simplement spécifier la taille d'une fenêtre. Cela définit la fenêtre par défaut, c'est-à-dire celle où l'affichage aura normalement lieu. Ici, la fenêtre par défaut est une simple ligne au bas de l'écran.

Lignes 2070-2120 : Ces lignes définissent deux autres fenêtres. L'une (#1) est une simple ligne en haut de l'ecran et l'autre (#2) est la partie principale de l'ecran. La commande SPEED INK est utilisée lorsqu'un numéro INK a deux couleurs qui lui sont affectées; elle spécifie la vitesse de clignotement des deux couleurs lors de l'affichage des lettres.

Ligne 2130 : Leur utilisation sera décrite dans les commentaires concernant la partie principale du programme.

Module 1.3.2 : Couleurs de l'écran à la mise en route

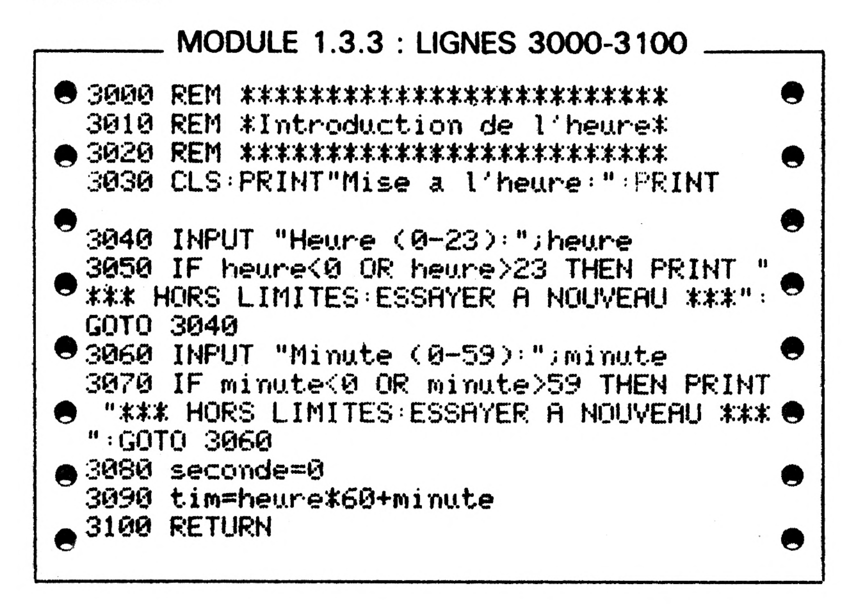
Nous avons dit dans les commentaires du dernier module, qu'il était possible de modifier les couleurs de l'écran pendant le déroulement du programme. Ce petit module définit les couleurs initiales.

MODULE 1.3.2 : LIGNES 1200-1280

```
12000 REM ************************
12010 REM *Préparation de l'ecran*
12020 REM ************************
12030 INK 0,1
12040 INK 1,24
12050 INK 2,26
12060 INK 3,2
12070 BORDER 1
12080 RETURN
```

Module 1.3.3 : Introduction de l'heure

Ce module est semblable à celui du programme Heurclassique, à la seule différence que nous adoptons ici une horloge de 24 heures.



MODULE 1.3.3 : LIGNES 3000-3100

```
3000 REM *************************
3010 REM *Introduction de l'heure*
3020 REM *************************
3030 CLS:PRINT"Mise a l'heure:":PRINT

3040 INPUT "Heure (0-23):";heure
3050 IF heure<0 OR heure>23 THEN PRINT "
*** HORS LIMITES:ESSAYER A NOUVEAU ***":
GOTO 3040
3060 INPUT "Minute (0-59):";minute
3070 IF minute<0 OR minute>59 THEN PRINT
 "*** HORS LIMITES:ESSAYER A NOUVEAU ***
":GOTO 3060
3080 seconde=0
3090 tim=heure*60+minute
3100 RETURN
```

Module 1.3.4 : Réglage de l'heure

Ce module est similaire à ses homologues précédents.

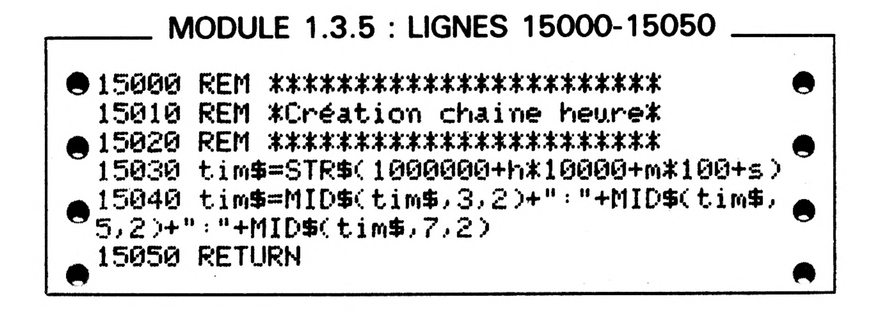
MODULE 1.3.4 : LIGNES 14000-14180

```
14000 REM ********************
14010 REM *Réglage de l'heure*
14020 REM ********************
14030 seconde=seconde+5
14040 WHILE seconde>59
14050 seconde=0:minute=minute+1
14060 WHILE minute>59
14070 minute=0:heure=heure+1
14080 WHILE heure>23
14090 heure=0
14100 WEND
14110 WEND
14120 WEND
14130 LOCATE#1,11,1
14140 h=heure:m=minute:s=seconde:GOSUB 1
5000
14150 PRINT#1,tim$;
14160 tim=heure*60+minute
14170 GOSUB 9000
14180 RETURN
```

Module 1.3.5 : Formatage de l'heure

Comme dans le programme précédent, le module de réglage de l'heure fonctionne conjointement à celui-ci, qui traduit l'heure fournie sous une forme intelligible par l'utilisateur.

MODULE 1.3.5 : LIGNES 15000-15050



```
15000 REM ***********************
15010 REM *Création chaine heure*
15020 REM ***********************
15030 tim$=STR$(1000000+h*10000+m*100+s)
15040 tim$=MID$(tim$,3,2)+":"+MID$(tim$,
5,2)+":"+MID$(tim$,7,2)
15050 RETURN
```

Module 1.3.6 : Test des compteurs et sonnerie

Ce module fait partie intégrante du programme car il permet à d'autres parties du programme de marquer une pause pendant laquelle l'utilisateur peut faire une entrée, tout en laissant le programme poursuivre la lecture des compteurs, pour déclencher les sonneries nécessaires. Nous allons entrer le module à ce stade parce qu'il constitue une sous-routine essentielle pour le module de menu suivant; toutefois, nous ne l'utiliserons pleinement qu'après avoir entré plusieurs autres modules.

MODULE 1.3.6 : LIGNES 9000-9300

```
9000 REM ******************
9010 REM *Test du compteur*
9020 REM ******************
9030 IF thoraire(0)<>tim THEN RETURN
9040 CLS #2:PRINT #2
9050 GOSUB 12000:INK 2,26,2
9060 PRINT #2,TAB(19);"SONNERIE";TAB(19)
;"========":LOCATE #2,1,11
9070 mlen=LEN(message$(0))
9080 PRINT #2,TAB(19-mlen/2);STRING$(mle
n+4,"*")
9090 PRINT #2,TAB(19-mlen/2);"*";TAB(22+
mlen/2);"*"
9100 PRINT #2,TAB(19-mlen/2);"* ";messa9
e$(0);" *"
9110 PRINT #2,TAB(19-mlen/2);"*";TAB(22+
mlen/2);"*"
9120 PRINT #2,TAB(19-mlen/2);STRING$(mle
n+4,"*")
9130 note=250
9140 WHILE INKEY$="" AND thoraire(0)=tim

9150 WHILE (SQ(1) AND 7)>0
9160 SOUND 1,note,15,15
9170 note=750-note
9180 WEND
9190 WEND
9200 sonné=1
9210 FOR i=1 TO compteurs-1
9220 theure(i-1)=theure(i)
9230 tminute(i-1)=tminute(i)
9240 thoraire(i-1)=thoraire(i)
9250 message$(i-1)=message$(i)
```

```
9260 NEXT i
9270 compteurs=compteurs-1
9280 IF compteurs=0 THEN thoraire(0)=-1
9290 INK 2,26
9300 RETURN
```

Commentaires

Ligne 9030 : C'est notre première référence pratique au tableau TTIME, qui stocke le moment d'expiration des compteurs. Les modules suivants, qui vous permettront de définir les compteurs, veilleront à ce que le premier compteur soit toujours placé sur la première ligne du tableau, l'élément 0. Le plus souvent, lorsque cette sous-routine est appelée, l'exécution se termine sur cette ligne. La sonnerie ne retentira que si l'heure actuelle, TIM (calculée par le module de réglage de l'heure), est identique à l'heure de sonnerie stockée sur la première ligne de TTIME.

Lignes 9040-9120 : Ces lignes paraissent plus complexes qu'elles ne le sont en réalité. Elles ont pour rôle d'afficher le mot 'SONNERIE' dans la fenêtre #2 (la partie principale de l'ecran), ainsi que tout message éventuel associé à la sonnerie particulière. Le message sera entouré d'astérisques et centré sur l'écran par l'affichage du premier caractère sur la position déterminée par la formule suivante :

POSITION CENTRALE SUR L'ECRAN - LONGUEUR DE LA CHAINE A AFFICHER/2

Vous pouvez voir cette formule sur les lignes 9090-9120, où elle prend la forme '19-mlen/2'.

Lignes 9130-9200 : Ces lignes font retentir (SOUND) une sonnerie à deux tons pendant une minute : durée pendant laquelle le temps d'expiration du compteur est identique à l'heure actuelle en minutes. La sonnerie peut être interrompue en actionnant une touche. La boucle intérieure joue la note réelle, la ligne 9150 assurant que chaque note est maintenue jusqu'à ce que la fonction SQ indique que la note précédente a cessé. La variable SONNE reçoit la valeur 1, pour indiquer au module de menu qu'une sonnerie a été déclenchée. Ce dernier point sera expliqué dans les commentaires sur le menu.

Lignes 9210-9280 : Après qu'une sonnerie ait retenti, ces lignes la suppriment des tableaux qui stockent les heures et les messages pour les compteurs. Pour cela, toutes les données du deuxième compteur sont déplacées d'une position vers le bas,

où elles détruisent le premier élément qui vient juste de se faire entendre. La variable COMPTEURS enregistre le nombre de compteurs actuellement définis, et elle a donc été réduite de 1. Enfin, s'il ne reste plus de compteurs, la ligne 9280 permet de placer une heure impossible à la position du premier compteur, pour qu'il ne sonne pas à n'importe quelle heure.

Module 1.3.7 : Le menu du programme

Et voici une nouvelle technique que nous utiliserons largement dans les programmes suivants : le 'menu'. Dans les programmes précédents, le program- me lui-même a assuré son propre contrôle. Dès le lancement (RUN), un module de contrôle a 'pris la main' et a commandé le déroulement du programme, jusqu'à ce que l'utilisateur indique la fin du programme.

Ce programme (et bon nombre des suivants) est différent car il n'y a pas qu'un déroulement unique du programme. L'utilisateur a diverses possibilités et c'est lui qui, dans une large mesure, dirige la manoeuvre. Pour cela, un module appelé module du programme, présente une liste de choix. Plus loin dans ce livre, nous trouverons des programmes plus complexes avec plusieurs menus, dont chacun offre à son tour plusieurs choix; pour le moment, nous nous en tiendrons au seul menu nécessaire à ce programme.

MODULE 1.3.7 : LIGNES 4000-4190

```
4000 REM ******
4010 REM *Menu*
4020 REM ******
4030 WHILE x$<>"5"
4040 CLS #2:PRINT #2
4050 PRINT #2,TAB(19);"MENU";TAB(19);"==
==":PRINT #2
4060 PRINT #2," 1) Initialiser le compte
ur"
4070 PRINT #2," 2) Supprimer le compteur
"
4080 PRINT #2," 3) Afficher les compteur
s et attendre"
4090 PRINT #2," 4) Effacer l'ecran et at
tendre"
4100 PRINT #2," 5) Arreter"
4110 sonné=0:x$=""
```

```
4120 WHILE x$="" AND sonne=0
4130 x$=INKEY$
4140 IF sonne THEN x$=""
4150 WEND
4160 ON VAL(x$) GOSUB 5000,6000,7000,800
0
4170 PRINT
4180 WEND
4190 RETURN
```

Commentaires

Lignes 4030-4170 : Cette boucle continue d'afficher le menu chaque fois que ce module reprend la main, jusqu'à ce que vous entriez le chiffre 5. Alors, l'exécution du programme revient au module de contrôle, que nous n'avons pas encore entré.

Lignes 4040-4100 : Ces lignes affichent la liste des options du programme dans la fenêtre #2, la feñetre principale qui occupe la plus grande partie de l'écran.

Lignes 4110-4150 : Habituellement, à ce stade du module de menu, il y aurait une instruction INPUT normale. Mais nous avons inclus un groupe de lignes plus complexes pour appeler le module précédent, car nous voulons lire continuellement l'état des compteurs, même pendant l'affichage du menu. Avec INPUT, ce serait impossible car une telle instruction bloque le programme jusqu'à ce que la touche ENTER soit actionnée. Même l'instruction EVERY n'a pas priorité sur INPUT.

Ces lignes définissent la variable qui sera utilisée pour stocker l'entrée de l'utilisateur (X$), sous la forme d'une chaîne vide, et pour continuer l'exécution de la boucle tant qu'elle reste vide. Si vous frappez une touche, cette action est détectée par l'instruction INKEY$ de la ligne 4130 et le caractère correspondant à la touche est stocké en X$, ce qui met fin à la boucle.

Toutefois, pendant cette lecture du clavier, le module précédent est constamment appelé pour voir si une sonnerie doit être déclenchée. La variable SONNE est utilisée de façon très simple. Normalement, lors de l'appel du module précédent, rien ne se passe et le menu peut continuer. Toutefois, si une sonnerie est déclenchée, le menu disparaît de l'écran et est remplacé par la sonnerie et par un message. Dans ce cas, SONNE recevra la valeur 1 pour indiquer au module de menu qu'il faut réafficher le menu sur l'écran.

Ligne 4140 : L'entrée faite par l'utilisateur est, dans la mesure du possible, transformée en un nombre par la fonction VAL. La commande ON...GOSUB choisit dans la liste des destinations celle qui correspond au nombre entré par l'utilisateur. Si le nombre entré est hors des limites de la liste des destinations de GOSUB, (0 ou supérieur à la longueur de la liste), la commande ON...GOSUB n'est pas exécutée et la boucle affiche à nouveau le menu.

Test

Avant de tester le programme, il vaut mieux entrer le petit module de contrôle suivant.

Module 1.3.8 : Le module de contrôle

Comme l'essentiel du travail est supporté par le menu, ce module est fort simple. On retrouve encore la synchronisation obtenue par une instruction EVERY incorporée dans ce module. Cette fois-ci, la synchronisation a lieu toutes les cinq secondes, comme nous l'avons indiqué dans les commentaires sur le module de réglage de l'heure.

MODULE 1.3.8 : LIGNES 1000-1080

```
1000 REM **********
1010 REM *Controle*
1020 REM **********
1030 GOSUB 3000
1040 EVERY 300 GOSUB 14000
1050 GOSUB 2000
1060 GOSUB 4000
1070 MODE 1
1080 END
```

Test

Si vous lancez le programme, vous devez voir apparaître le message demandant l'introduction de l'heure, suivi du menu. Entrez quelques nombres incorrects pour bien vérifier qu'ils ne sont pas acceptés. Si vous spécifiez les options 1 à 4 du programme, le programme s'arrête et un message d'erreur 'Line does not exist' (Cette ligne n'existe pas) apparaît, tandis que l'option 5 efface l'écran et met fin au programme.

Module 1.3.9 : Affichage des réglages des compteurs en vigueur

Ce module permet d'afficher de façon ordonnée, les heures réglées pour chacun des 16 compteurs. Pour l'instant, comme nous n'avons pas encore entré le module nous permettant de définir les valeurs des compteurs, l'affichage sera vierge ; mais, en entrant le module à présent, nous pourrons tester le module de réglage des compteurs, dès que nous l'aurons entré.

MODULE 1.3.9 : LIGNES 7000-7170

```
7000 REM ******************************
7010 REM *Afficher les compteurs      *
7015 REM *      et attendre           *
7020 REM ******************************

7030 k$=""
7040 WHILE k$=""
7050 CLS #2:PRINT #2
7060 PRINT #2,TAB(18);"COMPTEURS";TAB(18
);"=========":PRINT #2
7070 FOR i=0 TO compteurs-1
7080 h=theure(i):m=tminute(i):s=0:GOSUB
15000
7090 PRINT #2,USING " ##) &";i+1;tim$+"
"+message$(i)
7100 NEXT i
7110 sonne=0
7120 WHILE k$="" AND sonne=0
7130 k$=INKEY$
7140 IF sonne=1 THEN k$=""
7150 WEND
7160 WEND
7170 RETURN
```

Commentaires

Lignes 7030-7040 et 7160 : L'affichage persiste jusqu'à ce que vous frappiez une touche.

Lignes 7070-7100 : Comme nous l'avons vu, le nombre de compteurs actuellement réglés est enregistré dans la variable COMPTEURS. Cette boucle affiche les réglages du nombre de compteurs indiqué par COMPTEURS ; elle utilise le module

1.3.5 pour créer une présentation en chaîne de l'heure telle qu'elle est enregistrée dans les tableaux THEURE et TMINUTE, conjointement à PRINT USING pour obtenir un tableau ordonné. Le '&' dans l'instruction PRINT USING vous surprendra peut-être; il permet de mentionner un nombre, puis une chaîne dans la même instruction. S'il n'y avait pas de '&', une erreur 'Type mismatch' (Différence de type) apparaîtrait lorsque le programme essaierait d'interpréter TIM$ comme un nombre, à la ligne 7090.

Lignes 7110-7150 : Comme dans le menu, ces lignes représentent un état d'attente pendant lequel une commande EVERY lit le contenu des compteurs. A noter que si une sonnerie retentit pendant l'exécution de la boucle et que vous actionnez une touche pour arrêter la sonnerie, l'affichage des compteurs n'est pas interrompu. La ligne 7140 assure que, si la variable SONNE indique le déclenchement d'une sonnerie, il faut à nouveau actionner une touche avant que le programme ne revienne au menu.

Test

Tapez :

CLEAR :COMPTEURS=5 :GOTO 7000[ENTER]

L'heure des quatre compteurs doit s'afficher, bien que cette heure ne soit que '00-00-00' et qu'il n'y ait aucun message, puisqu'aucun n'a été entré. Si vous frappez une touche, le message d'erreur 'Unexpected RETURN' (RETURN inattendu) apparaît.

Module 1.3.10 : Introduction de l'heure pour un compteur absolu

Dans un instant, nous entrerons le module permettant de régler les compteurs. Toutefois, deux autres modules doivent le précéder, pour nous permettre d'entrer l'heure d'expiration d'un compteur, sous deux formes au choix. Lors du réglage d'un compteur, vous devez spécifier si l'heure est exprimée en heure absolue (c'est-à-dire la minute et l'heure à laquelle la sonnerie doit retentir), ou s'il s'agit d'un compte à rebours (c'est-à-dire si la sonnerie doit retentir *après* le nombre spécifié d'heures et de minutes). Le module actuel traite de l'entrée d'heures absolues et est similaire au module d'introduction de l'heure principale.

MODULE 1.3.10 : LIGNES 11000-11070

```
11000 REM ******************************
11010 REM *Introduction d'un compteur  *
11015 REM *          absolu            *
11020 REM ******************************
11030 INPUT "Heure (0 to 23):",theure
11040 IF theure<0 OR theure>23 THEN PRIN
T "**** HORS LIMITES:ESSAYER A NOUVEAU *
***";:FOR i=1 TO 2000:NEXT i:GOTO 11030
11050 INPUT "Minute (0 to 59):",tminute
11060 IF tminute<0 OR tminute>59 THEN PR
INT "**** HORS LIMITES:ESSAYER A NOUVEAU
 ****";:FOR i=1 TO 2000:NEXT i:GOTO 1105
0
11070 RETURN
```

Module 1.3.11 : Introduction de l'heure pour un compteur à rebours

Ce deuxième type de compteur est évidemment plus compliqué car, si dans le cas précédent nous indiquions l'heure de déclenchement de la sonnerie, ici le programme doit la calculer en ajoutant, la durée spécifiée à l'heure actuelle.

MODULE 1.3.11 : LIGNES 10000-10090

```
10000 REM ***************************
10010 REM *Introduction d'un compte*
          *       a rebours*       *
10020 REM ***************************
10030 INPUT "Durée en heures (0 to 23):"
,theure
10040 IF theure<0 OR theure>23 THEN PRIN
T "**** HORS LIMITES:ESSAYER A NOUVEAU *
***";:FOR i=1 TO 2000:NEXT i:GOTO 10030
10050 INPUT "Minutes to go (0 to 59):",t
minute
10060 IF tminute<0 OR tminute>59 THEN PR
INT "**** HORS LIMITES:ESSAYER A NOUVEAU
 ****";:FOR i=1 TO 2000:NEXT i:GOTO 1005
0
```

```
10070 tminute=tminute+minute:IF tminute>
59 THEN tminute=tminute-60:theure=theure
+1
10080 theure=theure+heure:IF theure>23 T
HEN theure=theure-24
10090 RETURN
```

Commentaires

Lignes 10070-10080 : La valeur actuelle des heures et des minutes est calculée par le module de réglage de l'heure qui sera appelé régulièrement par une instruction EVERY, lorsque la totalité du programme sera entrée. Les heures et les minutes indiquées par l'utilisateur (THEURE et TMINUTE) peuvent par conséquent être ajoutées aux variables HEURES et MINUTE.

Module 1.3.12 : Réglage des compteurs

A ce stade, nous avons désormais la possibilité de donner l'heure à un compteur. Nous pouvons passer au module principal qui permet d'initialiser les 16 compteurs. A noter que ce module utilise INPUT pour obtenir les trois données nécessaires; donc, pendant toute son exécution, il n'y a aucune lecture des compteurs, et par conséquent, aucune sonnerie ne sera déclenchée.

A noter également que vous ne pouvez pas faire attendre indéfiniment une réponse à une instruction INPUT. Pendant ce temps, le système enregistre le nombre de fois où il aurait exécuté la commande EVERY, éventuellement utilisée pour synchroniser le programme. Il peut même utiliser tout l'espace réservé à cet effet, et pour chaque instruction EVERY manquée, un appel du module de synchronisation sera oublié, de sorte que l'heure enregistrée par le programme sera incorrecte.

MODULE 1.3.12 : LIGNES 5000-5300

```
5000 REM ***********************
5010 REM *Réglage des compteurs*
5020 REM ***********************
5030 IF compteurs=16 THEN PRINT "     ***
***** PAS DE COMPTEURS LIBRES ********":
RETURN
5040 INPUT "Compte à rebours (o/n)";down
$
```

```
5050 IF LOWER$(down$)="o" THEN GOSUB 100
00 ELSE GOSUB 11000
5060 thoraire=theure*60+tminute
5070 rtim1=thoraire-tim:IF rtim1<=0 THEN
 rtim1=rtim1+1440
5080 FOR i=0 TO compteurs-1
5090 rtim2=ttime(i)-tim:IF rtim2<0 THEN
rtim2=rtim2+1440
5100 IF rtim1>rtim2 THEN NEXT i
5110 IF i<compteurs AND thoraire=thorair
e(i) THEN PRINT "    ****** COMPTEUR DEJ
A INITIALISE ******":FOR j=1 TO 2000:NEX
T:RETURN
5120 tnum=i
5130 message$=""
5140 WHILE message$=""
5150 INPUT "Message";message$
5160 IF LEN(message$)>25 THEN PRINT "***
 MESSAGE TROP LONG (25 CAR. MAX.) ***";:
FOR i=1 TO 2000:NEXT i:message$=""
5170 WEND
5180 PRINT
5190 FOR i=compteurs-1 TO tnum STEP -1
5200 theure(i+1)=theure(i)
5210 tminute(i+1)=tminute(i)
5220 thoraire(i+1)=thoraire(i)
5230 message$(i+1)=message$(i)
5240 NEXT i
5250 compteurs=compteurs+1
5260 theure(tnum)=theure
5270 tminute(tnum)=tminute
5280 thoraire(tnum)=thoraire
5290 message$(tnum)=message$
5300 RETURN
```

Commentaires

Ligne 5030 : Si le nombre de compteurs déjà initialisés est égal à 16, soit le maximum, un message d'erreur apparaît et le programme revient au menu.

Lignes 5040-5050 : Vous devez spécifier si un compte à rebours est nécessaire. La ligne 5050 traduit l'entrée, si besoin est, en majuscules afin que 'Y' ou 'y' soient acceptables, puis elle appelle l'un des deux modules précédents.

Lignes 5060-5070 : L'heure retournée par le module d'introduction de l'heure est convertie en un nombre représentant le nombre de minutes depuis le début du jour. Puis, il y a soustraction de l'heure actuelle en minutes. Si le résultat est négatif, l'heure spécifiée concerne le jour suivant et 1440 minutes (un jour, exprimé en minutes) sont ajoutées. Par exemple, si l'heure actuelle est 1830 (6.30 de l'après-midi) et si le compteur a été initialisé avec la valeur 1800 (6.00 de l'après-midi), le programme suppose que la sonnerie doit retentir à 1800 le jour suivant.

Lignes 5080-5120 : Cette boucle fait que le programme stocke toutes les initialisations de compteur dans l'ordre correct, de sorte que le compteur, réglé avec la première heure, soit aussi le premier de la liste. Pour cela, le nouvel élément doit être comparé aux heures des compteurs existants, pour situer sa place dans la liste.

Si le(s) premier(s) compteur(s) de la liste ont des heures inférieures à l'heure actuelle (autrement dit s'ils doivent se déclencher le jour suivant) ils se voient également ajouter 1440 minutes pour les besoins de la compa- raison. Le cas échéant, une heure peut être ultérieure au nouveau réglage du compteur et l'instruction IF de la ligne 5100 termine la boucle FOR. S'il n'y a pas de réglage actuel ultérieur au nouveau, la boucle se déroule jusqu'à ce que I soit égal à COMPTEURS, car, lorsque une boucle se termine naturellement, sa variable est incrémentée une dernière fois à la sortie de la boucle.

Donc, la valeur de I pointe actuellement, *soit* sur un réglage actuel de compteur ultérieur au nouveau réglage, *soit* sur le premier espace à la fin de la liste actuelle des compteurs. Un autre test est effectué, pour voir si la position indiquée a déjà un compteur ayant la même valeur que celle qui vient d'être entrée. Si c'est le cas, un message d'erreur apparaît et le programme revient au menu. A condition que cette erreur ne survienne pas, TNUM, l'endroit où le nouveau compteur doit être inséré, reçoit la valeur de I.

Lignes 5130-5170 : Le message à associer au compteur est demandé.

Lignes 5190-5240 : Comme nous savons où le nouveau compteur doit se trouver dans la liste, nous pouvons déplacer les compteurs existants d'une position vers le haut, à partir de cette position, créant ainsi un espace. Pour être enregistré, chaque compteur nécessite quatre informations qui se trouvent dans les tableaux THEURE, TMINUTE, TTIME et MESSAGE$.

Lignes 5250-5290 : La variable COMPTEURS est à présent incrémentée, indiquant qu'un autre compteur a été ajouté, et les quatre données nécessaires sont insérées dans les quatre tableaux, à la position indiquée par TNUM.

Test

A ce stade, vous êtes en mesure de tester le module en exécutant la totalité du programme. Lorsque le menu apparaît, choisissez l'option 1 et entrez :

Y,0,1 et TEST

en réponse aux quatre messages, après quoi le menu doit réapparaître. Ensuite, choisissez l'option 2 pour afficher les compteurs et vous constaterez que le compteur 0 a reçu une valeur quelque peu en avance sur l'heure actuelle (ce petit élément de temps est fonction de votre vitesse) plus le libellé 'TEST'. N'espérez pas la sonnerie, car nous n'avons pas encore entré le module à cet effet.

Module 1.3.13 : Effacement de l'écran

Ce programme est prévu pour une exécution permanente non prioritaire, pour jouer un rôle de synchronisation (avec arrêt et reprise automatiques). Le fait de laisser le CPC 464 sous tension pour cela est absolument inoffensif. Mais si vous laissez l'écran sous tension avec le même affichage pendant une longue durée, certaines des parties les plus brillantes de l'affichage peuvent être légèrement floues sur l'écran, et elles peuvent nuire à la lisibilité. (Pas d'affolement ! Cela ne se produira pas simplement parce que vous laissez le 464 et le moniteur sous tension pendant quelques heures par mégarde ; il s'agit plutôt d'un même affichage pendant une longue période).

Pour pallier cet éventuel problème, nous avons incorporé dans le programme un module de protection d'écran qui efface ce dernier jusqu'à ce qu'une touche soit actionnée. Pendant que l'écran s'efface, les compteurs sont lus en permanence, et toutes les sonneries seront déclenchées normalement.

L'effacement de l'écran par ce module est obtenu en donnant à toutes les encres (INK) la couleur de fond. Le module suivant appelle ce module et crée un état d'attente jusqu'à la réactivation de l'écran.

MODULE 1.3.13 : LIGNES 13000-13080

```
13000 REM ***********************
13010 REM *Effacement de l'ecran*
13020 REM ***********************
13030 INK 0,0
13040 INK 1,0
13050 INK 2,0
13060 INK 3,0
13070 BORDER 0
13080 RETURN
```

Module 1.3.14 : Etat d'attente avec un écran vierge

Ce module n'est rien d'autre qu'une boucle de temporisation semblable à celles des modules d'affichage du menu et des compteurs; toutefois, l'appel du module provoque l'effacement de l'écran et, lorsqu'une touche est actionnée, elle réinstaure les couleurs de départ.

MODULE 1.3.14 : LIGNES 8000-8130

```
8000 REM ***********************
8010 REM *Effacement de l'ecran*
8020 REM ***********************
8030 k$=""
8040 WHILE k$=""
8050 GOSUB 13000
8060 sonne=0
8070 WHILE k$="" AND sonne=0
8080 k$=INKEY$
8090 IF sonne=1 THEN k$=""
8100 WEND
8110 WEND
8120 GOSUB 12000
8130 RETURN
```

Test

Réglez un ou deux compteurs avec de courtes périodes, puis choisissez l'effacement de l'écran sur le menu. Bien que vous ne puissiez pas voir l'état des compteurs, vous constaterez que les sonneries sont déclenchées normalement.

Module 1.3.15 : Suppression d'un compteur

Parfois, la valeur d'un compteur doit être supprimée ou modifiée. Ce module vous permet de supprimer un compteur. La technique de base utilisée ici est importante, bien que simple, pour les programmes de manipulation de données. Elle consiste à décaler vers le bas le fichier pour 'écraser' le compteur à supprimer.

MODULE 1.3.15 : LIGNES 6000-6140

```
6000 REM ***********************
6010 REM *Supprimer le compteur*
6020 REM ***********************
6030 IF compteurs=0 THEN PRINT "    ****
**** PAS DE COMPTEURS INITIALISES*******
*":FOR i=1 TO 2000:NEXT:RETURN
6040 INPUT "Quel compteur";tnum
6050 IF tnum>compteurs THEN PRINT "    *
******* COMPTEUR INCONNU*********":FOR i=
1 TO 2000:NEXT:RETURN
6060 FOR i=tnum TO compteurs-1
6070 theure(i-1)=theure(i)
6080 tminute(i-1)=tminute(i)
6090 thoraire(i-1)=thoraire(i)
6100 message$(i-1)=message$(i)
6110 NEXT i
6120 compteurs=compteurs-1
6130 IF compteurs=0 THEN thoraire(0)=0
6140 RETURN
```

Commentaires

Lignes 6030-6050 : Messages d'erreur si le numéro de compteur entré n'existe pas.

Lignes 6060-6110 : 'L'écrasement' : tout ce qui se trouve au-dessus de l'élément à supprimer est décalé d'un espace vers le bas.

Test

Vous pouvez à présent initialiser un compteur puis, avec l'option 2 du menu, le supprimer avant qu'il ne sonne. Si ce test est satisfaisant, le programme est opérationnel.

PROGRAMME 1.4 : EVENEMENT

Fonction du programme

Le programme final de ce chapitre d'expériences sur l'heure est plutôt inhabituel. En effet, il essaie de transformer le 464 en chronomètre. Bien sûr, la précision d'un micro-ordinateur et celle de chronomètres spécialisés ne sont pas comparables, mais l'ordinateur présente certains avantages : il peut garder une trace des heures qu'il a produites, effectuer des calculs sur ces heures, et autres possibilités. Ce programme est conçu pour chronométrer un ou plusieurs événements, pour afficher les différents temps, avec ou sans message d'identification, et, à chaque événement, pour calculer la différence de temps avec le précédent. Il peut aussi, à la demande, faire une sortie imprimée de la liste des temps.

```
11:15:04 (00:00:04)
11:15:05 (00:00:01)
11:15:08 (00:00:03)
11:15:15 (00:00:07)AUTOBUS
11:15:24 (00:00:09)
11:15:25 (00:00:01)
11:15:27 (00:00:02)
11:15:27 (00:00:00)
11:15:35 (00:00:08)CAMION
11:15:37 (00:00:02)
11:15:38 (00:00:01)
11:15:40 (00:00:02)
11:15:48 (00:00:08)FOURGONNETTE
11:15:50 (00:00:02)
11:15:51 (00:00:01)
11:15:54 (00:00:03)
11:15:57 (00:00:03)
11:16:01 (00:00:05)CHAR
```

Figure 1.4 : Liste du programme Evénement.

Dans ce chapitre, les points suivants sont introduits :

1) Calcul de temps et de durées.
2) Sortie imprimée.

Module 1.4.1 : Initialisation

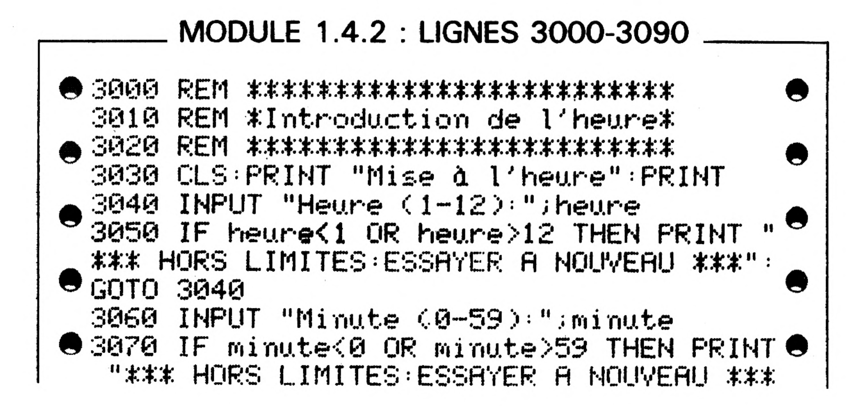
C'est un module d'initialisation standard qui définit une petite fenêtre par défaut au bas de l'écran, une fenêtre d'une ligne (# 1) en haut de l'ecran et une deuxieme (# 2) qui occupe la plus grande partie de l'écran.

MODULE 1.4.1 : LIGNES 2000-2160

```
2000 REM ****************
2010 REM *Initialisation*
2020 REM ****************
2030 MODE 1
2040 INK 0,1
2050 INK 1,24
2060 INK 2,26
2070 INK 3,2
2080 BORDER 1
2090 WINDOW 1,40,25,25
2100 PAPER 0:PEN 1:CLS
2110 WINDOW #1,6,35,1,1
2120 PAPER #1,0:PEN #1,2:CLS #1
2130 PRINT #1,"Heure actuelle:"
2140 WINDOW #2,1,40,3,23
2150 PAPER #2,3:PEN #2,2:CLS #2:PRINT #2
2160 RETURN
```

Module 1.4.2 : Réglage de l'heure

Le module standard de réglage de l'heure des deux derniers programmes.



MODULE 1.4.2 : LIGNES 3000-3090

```
3000 REM *************************
3010 REM *Introduction de l'heure*
3020 REM *************************
3030 CLS:PRINT "Mise à l'heure":PRINT
3040 INPUT "Heure (1-12):";heure
3050 IF heure<1 OR heure>12 THEN PRINT "
*** HORS LIMITES:ESSAYER A NOUVEAU ***":
GOTO 3040
3060 INPUT "Minute (0-59):";minute
3070 IF minute<0 OR minute>59 THEN PRINT
 "*** HORS LIMITES:ESSAYER A NOUVEAU ***
```

```
":GOTO 3060
3080 seconde=0
3090 RETURN
```

Module 1.4.3 : Attente d'entrée

Là aussi, nous utilisons INKEY$ pour éviter les problèmes liés à l'instruction EVERY lorsqu'INPUT est utilisé. Outre la création de l'état d'attente, le module peut accepter une instruction qui envoie la future sortie à une imprimante.

MODULE 1.4.3 : LIGNES 4000-4180

```
4000 REM *********
4010 REM *Attente*
4020 REM *********
4030 t$="":message$=""
4040 WHILE t$=""
4050 WHILE t$=""
4060 t$=INKEY$
4070 WEND
4080 WHILE LOWER$(t$)="p"
4090 imprimante=1-imprimante
4100 t$=""
4110 WEND
4120 WHILE t$=CHR$(13)
4130 INPUT "Message (<=18 car.):",messag
e$
4140 IF LEN(message$)<=18 THEN t$="*"
4150 WEND
4160 CLS
4170 WEND
4180 RETURN
```

Commentaires

Ligne 4030 : Cette chaîne est utilisée pour stocker toute touche actionnée par l'utilisateur.

Lignes 4040-4070 : Boucle principale, qui se déroule jusqu'à ce que T$ reçoive une valeur, soit par le programme, soit par le clavier.

Lignes 4050-4070 : Etat d'attente proprement dit, qui se poursuit jusqu'à ce qu'une touche soit actionnée.

Lignes 4080-4110 : Si la touche actionnée est celle de la lettre P, la valeur de la variable imprimante passe de 0 à 1. Retenez bien cette simple ligne, car elle représente la manière courante de faire basculer une variable entre deux valeurs. Si vous avez une variable X et si vous voulez la faire basculer des valeurs V1 à V2 et inversement (où V1 est la plus basse), voici le format utilisé : X = V1 + V2 - X

mais, comme notre valeur la plus basse est 0, nous simplifions ainsi : X = V2 - X

Lignes 4120-4150 : Si la touche actionnée est ENTER, la ligne du bas de l'écran vous invite à entrer un bref message qui accompagne l'heure de l'événement, après quoi cette ligne de commande est effacée à nouveau.

Test

Après vous être assuré que le programme est initialisé, tapez :

GOTO 4000[ENTER]

Le 464 se met en état d'attente. Si vous frappez la touche P, il n'y a pas d'action visible. Si vous appuyez sur [ENTER], un message vous est demandé et l'attente est terminée. Toute autre touche met simplement fin au module.

Module 1.4.4 : Modification de l'heure et du temps

Ce module est similaire à ceux des deux programmes précédents mais, outre la mise à jour constante de l'heure principale, il met à jour une deuxième forme de temps qui représente la période écoulée depuis la dernière action sur une touche. Cette deuxième forme de temps est stockée sous la forme de variables commençant par la lettre P.

MODULE 1.4.4 : LIGNES 6000-6280

```
6000 REM **************************
6010 REM *Modification de l'heure*
6015 REM *        et du temps     *
6020 REM **************************
6030 seconde=seconde+1
6040 WHILE seconde>59
6050 seconde=0:minute=minute+1
6060 WHILE minute>59
6070 minute=o:heure=heure+1
```

```
6080 WHILE heure>12
6090 heure=1
6100 WEND
6110 WEND
6120 WEND
6130 Pseconde=Pseconde+1
6140 WHILE Pseconde>59
6150 Pseconde=0:Pminute=Pminute+1
6160 WHILE Pminute>59
6170 Pminute=0:Pheure=Pheure+1
6180 WHILE Pheure>12
6190 Pheure=1
6200 WEND
6210 WEND
6220 WEND
6230 LOCATE#1,11,1
6240 h=heure:m=minute:s=seconde:GOSUB 70
00
6250 PRINT#1,tim$;
6260 h=Pheure:m=Pminute:s=Pseconde:GOSUB
 7000
6270 PRINT#1,"  (";tim$;")"
6280 RETURN
```

Test

Vous ne pouvez tester réellement ce module qu'après avoir entré le suivant, qui permet l'impression.

Module 1.4.5 : Affichage des résultats

Ce module, comme dans les deux programmes précédents, crée une chaîne à partir de l'heure actuelle.

MODULE 1.4.5 : LIGNES 7000-7050

```
7000 REM *************************
7010 REM *Création chaine d'heure*
7020 REM *************************
7030 tim$=STR$(1000000+h*10000+m*100+s)
7040 tim$=MID$(tim$,3,2)+":"+MID$(tim$,5
,2)+":"+MID$(tim$,7,2)
7050 RETURN
```

Test

Tapez :

CLEAR :CLS :GOTO 6000[ENTER]

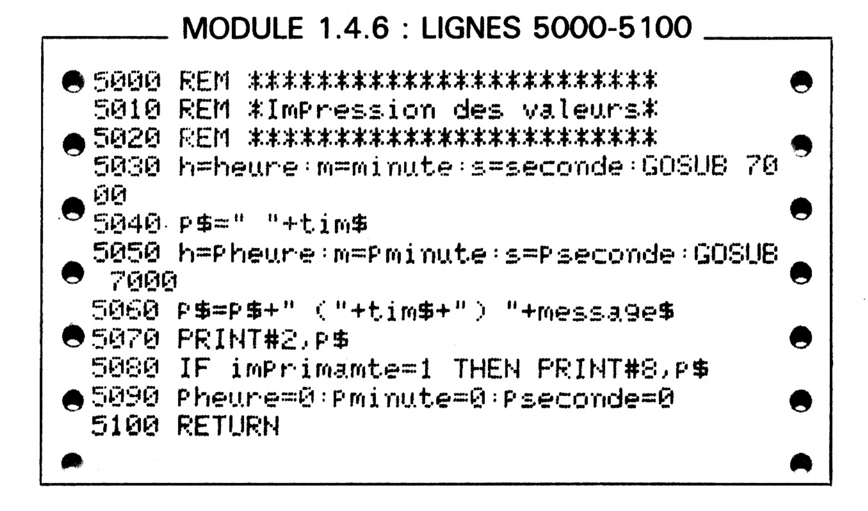
Vous devez voir :

00 :00 :00 (00 :00 :01)

affiché en haut de l'écran. Si vous entrez GOTO 6000 plusieurs fois, vous pourrez constater que l'heure augmente chaque fois d'une seconde.

Module 1.4.6 : Impression des résultats

Après avoir entré les modules d'attente et de mise à jour de l'heure, nous voulons à présent pouvoir imprimer le temps calculé lorsqu'une touche est actionnée.



MODULE 1.4.6 : LIGNES 5000-5100

```
5000 REM ************************
5010 REM *Impression des valeurs*
5020 REM ************************
5030 h=heure:m=minute:s=seconde:GOSUB 70
00
5040 p$=" "+tim$
5050 h=pheure:m=pminute:s=pseconde:GOSUB
 7000
5060 p$=p$+" ("+tim$+") "+message$
5070 PRINT#2,p$
5080 IF imprimamte=1 THEN PRINT#8,p$
5090 pheure=0:pminute=0:pseconde=0
5100 RETURN
```

Commentaires

Lignes 5030-5070 : Une chaîne est créée, constituée de l'heure actuelle (HEURE, MINUTE, SECONDE), suivie de la durée de la période (PHEURE, PMINUTE, PSECONDE) et de tout message entré par l'utilisateur. L'affichage a lieu sur la fenêtre principale de l'écran (# 2).

Ligne 5080 : Contrairement à beaucoup d'autres micros, le 464 maintient en permanence un canal de communication ouvert

avec toute imprimante connectée. Sur la plupart des autres micros, il faudrait 'ouvrir un fichier pour l'imprimante', soit plusieurs lignes de programmation BASIC. Sur le 464, il suffit d'envoyer ce que nous voulons imprimer au numéro de canal 8, toujours relié directement au port de l'imprimante.

Ligne 5090 : Les valeurs de période (temps écoulé) sont mises à zéro, car elles sont destinées à représenter le temps écoulé depuis la dernière action sur une touche.

Module 1.4.7 : Le module de contrôle

Pour rassembler le tout, voici le module de contrôle.

MODULE 1.4.7 : LIGNES 1000-1110

```
1000 REM **********
1010 REM *Controle*
1020 REM **********
1030 ON BREAK GOSUB 1110
1040 GOSUB 3000
1050 EVERY 50 GOSUB 6000
1060 GOSUB 6000
1070 WHILE 1
1080 GOSUB 4000
1090 GOSUB 5000
1100 WEND
1110 CLEAR:MODE 1:END
```

Commentaires

Ligne 1070 : Un petit détail qui peut vous être utile : le '1' après l'instruction WHILE permet le déroulement indéfini de cette boucle. Toute autre valeur positive donnerait le même résultat.

Conclusion

Vous pouvez tirer un certain nombre d'enseignements des programmes de ce chapitre. Tout d'abord, le fait que l'usage de votre 464 n'est limité que par votre imagination. Mais le point le plus important à retenir, si vous revoyez les programmes précédents, est la facilité de conception des programmes si tout est découpé en modules bien distincts, transférables d'un programme à un autre. Avec sa commande MERGE facile d'emploi,

le 464 demande impérativement à être utilisé ainsi. Lorsque vous aurez créé une bibliothèque suffisante de routines utiles, vous verrez qu'une bonne partie de la programmation consiste à rassembler les pièces d'un puzzle. Mais ce puzzle-là sera bien souvent plus utile que les précédents.

2

Des Dessins et des Nombres

Après avoir joué avec l'heure et le temps, nous allons dans ce chapitre voir quelque chose que votre 464 fait particulièrement bien : afficher l'information d'une façon plus parlante et plus compréhensible que des listes de faits et de chiffres. Vous trouverez trois programmes qui créent des graphiques de différents types, en basse et haute résolution.

Dans ce chapitre ainsi que dans les suivants, les explications iront en diminuant, sauf lorsqu'il s'agira de nouvelles idées ou de modules complexes. C'est volontaire ; nous avons voulu un équilibre entre l'information nécessaire à la compréhension des programmes, et les programmes proprement dits, qui sont votre raison d'achat de ce livre. Si un point vous paraît difficile, vous constaterez souvent qu'une technique similaire a été utilisée dans un programme précédent. Il vous suffira de relire le commentaire approprié. C'est pour cela que nous vous avons conseillé dans l'introduction de suivre l'ordre du livre, sans passer directement aux programmes les plus complexes.

Voici les programmes de ce chapitre :

COURBE : Crée un graphique de grande qualité sous forme de courbe en haute résolution.

CERCLE : Prend un petit nombre de données et les affiche sous la forme d'un cercle divisé en segments multicolores.

HISTOGRAMME 3D : Vous permet de créer un histogramme en trois dimensions tout-à-fait remarquable.

PROGRAMME 2.1 : COURBE

Fonction du programme

Dans les programmes de traitement des données, la façon d'entrer les données représente en général une consommation importante de temps et de mémoire. Cela se traduit souvent par des sections de programme longues et complexes, consacrées à la saisie, à la modification et à la suppression de données.

Dans les deux premiers programmes de ce chapitre, nous pallions ce problème en créant un programme qui, bien que facile d'emploi, n'utilise pas de mémoire pour afficher des messages sur l'écran, stocker des données dans les tableaux ou enregistrer des données sur cassette. Les programmes interactifs, c'est-à-dire ceux qui guident l'utilisateur pendant l'exécution, sont certes séduisants mais quelque peu luxueux lorsque la mémoire est limitée. Dans ce programme, qui trace un graphique haute résolution, nous utilisons des instructions DATA qui contribuent à la simplicité ; son écriture est pourtant bien plus simple que celle d'un programme interactif pour un résultat identique.

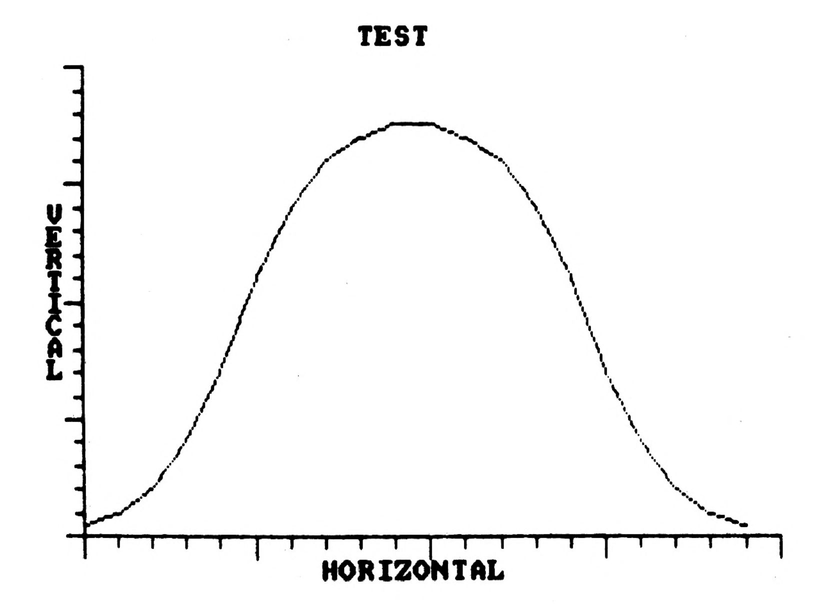


Figure 2.1 : Affichage obtenu avec le programme Courbe.

Modules 2.1.1 et 2.1.2 : Les données de la courbe

C'est sur ces deux modules que repose la simplicité du programme. Vous trouverez dans ces quelques lignes toutes les informations nécessaires au tracé d'une courbe, avec des libellés clairs, pour une création facile de l'affichage. Si, à ce stade, vous ne saisissez pas bien la signification de tous les chiffres, ne soyez pas inquiet; vous les comprendrez mieux lorsque vous aurez entré et exécuté le programme une ou deux fois, et que vous aurez essayé certaines modifications.

MODULE 2.1.1 et 2.1.2 : LIGNES 3000-4050

```
3000 REM **********
3010 REM *Donnée 1*
3020 REM **********
3030 DATA TITRE DE LA COURBE,TEST
3040 DATA NOM AXE/H,HORIZONTAL
3050 DATA NOM AXE/V,VERTICAL
3060 DATA UNITES SUR AXE/H,20
3070 DATA UNITES SUR AXE/V,20
3080 DATA VALEUR PAR UNITE/V,10
4000 REM **********
4010 REM *Donnée 2*
4020 REM **********
4030 DATA 5,10,20,40,70,110,140,160,170,
175
4040 DATA 175,170,160,140,110,70,40,20,1
0,5
4050 DATA END
```

Commentaires

Lignes 3030-3050 : Nom général donné à la courbe par l'utilisateur et libellés associés à l'axe vertical (ordonnées) et à l'axe horizontal (abscisses). A noter que, dans chaque cas, la phrase précédant la virgule dans l'instruction DATA n'est là que pour la clarté et qu'elle est ignorée par le programme. En fait, la virgule est essentielle pour séparer la partie explicative à gauche, de la partie d'information proprement dite à droite.

Lignes 3060-3070 : Les deux axes de la courbe sont divisés en unités, pour faciliter la lecture. Bien que ces axes aient toujours la même longueur, vous pouvez spécifier la division en un certain nombre d'unités.

Ligne 3080 : Si, par exemple, la courbe doit représenter des tonnes de céréales produites par un pays pendant un certain nombre d'années, vous pouvez décider que chaque unité de l'axe des ordonnées représente 1 000 tonnes. Plutôt que de vous demander de diviser le nombre réel en unités de 1000 avant la saisie, le chiffre de cette ligne permet de spécifier les unités, afin que chaque nombre puisse être entré intégralement.

Lignes 4030-4040 : Données de base de la courbe. Dans le présent programme, ces nombres donneront une courbe régulière en forme de cloche. A noter qu'un seul nombre par unité est prévu sur l'axe des abscisses, à partir de la position 1, bien que la totalité de cet axe ne soit pas utilisée. Vous pouvez placer autant de données que vous le souhaitez sur chaque ligne séparée.

Ligne 4050 : Le nombre d'instructions DATA n'est limité que par la mémoire, mais vous devez terminer par une instruction DATA contenant le mot END. Cela indique au programme qu'il a atteint la fin des données utilisées pour la courbe, même si d'autres instructions DATA suivent.

Module 2.1.3 : Traçage de la grille de la courbe

Ce module trace la grille quadrillée qui doit recevoir la courbe, avec les unités sur les axes et les divers libellés.

MODULE 2.1.3 : LIGNES 1000-1190

```
1000 REM ***********************
1010 REM *Traçage de la grille*
1020 REM ***********************
1030 MODE 1
1040 MOVE 32,368
1050 DRAW 32,32,2
1060 DRAW 639,32
1070 RESTORE 3000
1080 READ t$,t$:LOCATE 20-LEN(t$)/2,1:PR
INT t$
1090 READ t$,t$:LOCATE 21-LEN(t$)/2,25:P
RINT t$
1100 READ t$,t$:FOR i=1 TO LEN(t$):LOCAT
E 1,12-LEN(t$)/2+i:PRINT MID$(t$,i,1):NE
XT i
```

```
1110 READ t$,nv:lv=338/nv
1120 FOR i=o TO nv
1130 MOVE 24+8*(i/5=INT(i/5)),32+i*lv:DR
AW 32,32+i*lv
1140 NEXT i
1150 READ t$,nh:lh=606/nh
1160 FOR i=0 TO nh
1170 MOVE 32+i*lh,24+8*(i/5=INT(i/5)):DR
AW 32+i*lh,32
1180 NEXT i
1190 READ t$,unité
```

Commentaires

Lignes 1040-1060 : Les deux axes sont tracés, une ligne partant de l'angle supérieur gauche de l'écran et se dirigeant vers l'angle inférieur gauche, se poursuivant par un angle droit le long du bas de l'écran, vers l'angle inférieur droit.

Ligne 1070 : Le 'pointeur de données' du 464 pointe sur la première donnée qui suit le début du module à la position 3000. Cette instruction RESTORE empêche l'apparition d'une erreur 'DATA exhausted' (DATA épuisées) si le programme commence par GOTO. De toute façon, l'utilisation de RUN place le pointeur sur la première donnée.

Lignes 1080-1100 : Les libellés de l'ensemble du graphique, de l'axe des abscisses et de l'axe des ordonnées sont lus à partir des instructions DATA et affichés sur l'écran. Dans le cas de l'axe des ordonnées, une boucle permet d'afficher le libellé caractère par caractère verticalement sur la partie gauche de l'écran. A noter que dans chaque cas, il y a deux instructions READ. La première prend la phrase *avant* la virgule de l'instruction DATA. Cette lecture est immédiatement annulée par la lecture (READ) d'une autre chaîne dans la même variable, T$.

Ligne 1110 : Le nombre de graduations à placer sur l'axe des ordonnées (NV) est lu à partir de l'instruction DATA suivante. Ensuite, la longueur de l'axe des ordonnées (338 pixels) est divisée par ce nombre pour obtenir la longueur de chaque division en pixels (LV).

Lignes 1120-1140 : Cette boucle trace de petites graduations sur l'axe des ordonnées, pour représenter les divisions spécifiées par l'utilisateur. A noter particulièrement l'expression 8*(I/5=INT(I/5)), qui a pour effet de tracer une marque plus

importante toutes les cinq graduations. Pour comprendre cette expression, vous devez avoir des notions de traitement des 'conditions logiques' par le 464.

Lorsque l'interpréteur BASIC du 464, le programme en code machine qui exécute BASIC, rencontre une condition après une instruction IF, par exemple 'A>B', 'A=B' ou 'A<=B', il doit déterminer si cette condition est vraie ou fausse pour décider s'il exécute l'action spécifiée par l'instruction IF. Ainsi :

```
IF A>B THEN GOSUB 1000
```

sera exécutée si 'A>B' est *vrai* (par exemple A=10 et B=9) et ignorée si elle est fausse (par exemple A=9 et B=10). Pour la décision, la condition est évaluée d'après les valeurs en cours A et B (ou toute variable spécifiée) ; elle reçoit une valeur de -1 si elle est vraie, et de 0 si elle est fausse. C'est la *valeur* bien plus que la condition elle-même qui est importante, un fait facile à démontrer avec le petit test suivant. Entrez en mode direct :

```
A=5[ENTER]
IF A THEN PRINT «VRAI»[ENTER]
```

Résultat : «VRAI» s'affichera sur l'écran, car la valeur qui suit l'instruction IF n'est pas 0 (toute valeur différente de zéro, positive ou négative, produirait le même effet). A présent, essayez :

```
A=0[ENTER]
IF A THEN PRINT «VRAI»[ENTER]
```

Cette fois-ci, rien ne s'affiche. Toutefois pour l'instant, nous ne nous intéressons pas particulièrement à la façon dont IF fonctionne, mais à la manière d'évaluer les conditions. Donc, essayez ce qui suit :

```
A=1[ENTER]
B=1[ENTER]
PRINT A=B[ENTER]
```

Vous devez voir '-1', la valeur d'une condition vraie. Puis, essayez :

```
A=1[ENTER]
B=2[ENTER]
PRINT A=B[ENTER]
```

Le résultat sera '0' car la condition n'est pas vraie. Au début, cela peut paraître intéressant mais pas forcément utile. En fait, cette faculté d'extraire une valeur d'une condition logique est très importante en programmation, comme l'illustre la ligne 1130.

La ligne 1130 trace une série de lignes à angle droit sur l'axe des ordonnées, pour matérialiser les divisions spécifiées par l'utilisateur. Ces lignes ont normalement une longueur de 8 pixels. Toutefois, chaque fois que la valeur de la variable I de la boucle est exactement divisible par 5 (c'est-à-dire toutes les cinq marques), la condition (I/5=INT(I/5)) est vraie et amène la valeur -1 plutôt que 0. Autrement dit, en incluant '-8*(I/5=INT(I/5))' dans la ligne qui spécifie la longueur de la marque à tracer (DRAW), nous pouvons doubler la longueur de chaque 5ème marque, sans avoir recours à une instruction IF...THEN...ELSE complexe.

A noter que pour *ajouter* huit à la longueur de la marque, nous devons *retirer* huit fois la valeur de la condition, car sa valeur est de *moins* un si la réponse est vraie : soustraire un nombre négatif équivaut à en ajouter un positif.

Lignes 1150-1180 : La même opération a lieu pour l'axe des abscisses.

Ligne 1190 : Le nombre d'unités représenté par chaque division sur l'axe des ordonnées est lu à partir de l'instruction DATA.

Test

Pour tester cette partie du programme, il nous suffit d'exécuter le programme dans son état actuel. La grille de la courbe va apparaître sur l'écran avec 'TEST' en haut, 'VERTICAL' sur la partie gauche et 'HORIZONTAL' au bas de l'écran. Les deux axes sont divisés en 20 segments.

Module 2.1.4 : Traçage de la courbe

Après avoir défini le cadre, il nous reste à tracer la courbe proprement dite en utilisant les informations spécifiées dans le module DATA2.

MODULE 2.1.4 : LIGNES 2000-2160

```
2000 REM **********************
2010 REM *Traçage de la courbe*
2020 REM **********************
2030 ON ERROR GOTO 2140
2040 RESTORE 4000
2050 READ t
2060 PLOT 32,32+t/unité*lv,1
```

```
2070 colonne=1
2080 READ t$
2090 IF t$="FIN" THEN GOTO 2130
2100 DRAW 32+lh*colonne,32+VAL(t$)/unité
*lv
2110 colonne=colonne+1
2120 GOTO 2080
2130 IF INKEY$="" THEN GOTO 2130
2140 CLS
2150 LIST 3000-
2160 END
```

Commentaires

Lignes 2030 et 2130-2160 : Si une erreur survient pendant le traçage principal, ou si une touche est actionnée après la représentation de la courbe, l'écran s'efface et donne la liste des données ayant servi de base à la courbe. Il est donc très facile d'examiner la courbe et d'en changer certains détails.

Ligne 2050 : La première donnée est lue.

Ligne 2060 : La courbe commence sur l'axe des ordonnées, sur un point représentant la taille de la première donnée, exprimée en unités telles que vous les avez spécifiées;

Ligne 2070 : COLONNE permet d'enregistrer le nombre de données lues et, en conséquence, la distance de progression de la courbe sur l'axe des abscisses

Lignes 2080-2090 : Les données suivantes sont lues sous forme de chaînes. Ainsi, il est possible de tester si la donnée lue est le mot 'END' et, si c'est le cas, de terminer le programme.

Lignes 2100-2120 : Comme le curseur graphique se trouve déjà à l'extrémité de la partie de la courbe déjà tracée, il suffit de tracer (DRAW) jusqu'au point adéquat suivant, en incrémentant COLONNE pour chaque donnée lue. Les coordonnées X ou horizontales sont calculées en multipliant COLONNÉ (le nombre d'unités d'avancement de la courbe sur l'axe des abscisses) par la longueur en pixels des unités horizontales (LH) - la constante 32 représente la distance entre le début de l'axe des abscisses et la partie gauche de l'écran. Les coordonnées Y ou verticales sont d'abord divisées par UNITE.
Donc, si la donnée était de 1 000 000 et si vous aviez spécifié que l'axe des ordonnées devait être divisé en unités de 100 000

(UNITE = 100 000), le résultat serait de 1 000 000/100 000, soit 10 unités. Le nombre d'unités ainsi obtenu est ensuite multiplié par la longueur en pixels des unités verticales (LV).

Test

Exécutez le programme terminé et vous allez voir apparaître une courbe régulière en forme de cloche. Lorsque la courbe est terminée, frappez sur une touche pour afficher les modules DATA, afin de les modifier à votre gré. Si ce test est satisfaisant, le programme est prêt à l'emploi.

PROGRAMME 2.2 : CERCLE

Fonction du programme

Pour représenter de petites quantités de données, une technique convient particulièrement : le cercle (vous trouverez aussi les désignations : diagramme circulaire, tarte, camembert, etc.). Un cercle représentant un total est fragmenté en segments ou en secteurs correspondant aux différents composants du total. Dans le programme suivant, nous allons mettre en pratique nos connaissances sur les cercles et leur aspect mathématique, et sur l'utilisation souple des instructions DATA du dernier programme. En cas de doute, revoyez les programmes précédents, car nous n'expliquerons pas à nouveau ces techniques.

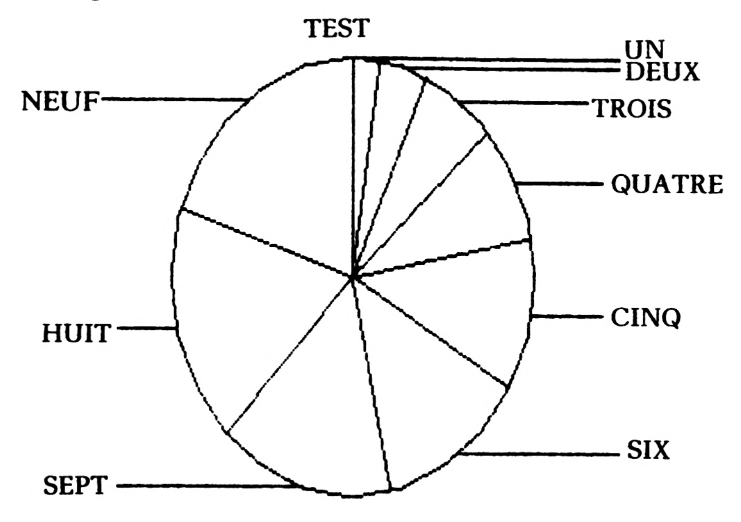


Figure 2.2 : Affichage obtenu avec Cercle.

Nous pouvons conserver les lignes entrées pendant le test, car elles doivent faire partie du dernier module, le module de contrôle.

Module 2.2.1 : Les données du cercle

Comme avec le graphique haute résolution précédent, les nombres servant de base au présent graphique sont contenus dans des instructions DATA qui se passent d'explication. A noter toutefois que dans ce programme particulier, les deux tableaux utilisés pour contenir le nom de chaque donnée et sa valeur ne sont pas dimensionnés ; vous êtes donc limité à 10 données. En fait, un diagramme circulaire de plus de 10 données présente peu d'intérêt, car trop chargé. Dans ce cas, il vaut mieux choisir une autre représentation graphique. Toutefois, vous pouvez, si vous le souhaitez, placer une instruction de dimensionnement au début du programme pour augmenter le nombre de données utilisables.

MODULE 2.2.1 : LIGNES 4000-4080

```
4000 REM ************************
4010 REM *Données pour le cercle*
4020 REM ************************
4030 DATA TITRE,TEST
4040 DATA NOMBRE D'ITEMS,9
4050 DATA NOM,UN,DEUX,TROIS,QUATRE,CINQ,
SIX,SEPT,HUIT,NEUF,DIX
4060 DATA
4070 DATA QUANTITES,1,2,3,4,5,6,7,8,9,10
4080 DATA
```

Module 2.2.2 : Traitement des données du cercle

Les informations du module DATA sont lues dans les variables NOM$ et DONNEES et dans les tableaux NOM$ et A.

MODULE 2.2.2 : LIGNES 5000-5110

```
5000 REM ************************
5010 REM *Traitement des données*
5020 REM ************************
5030 RESTORE 4000
5040 READ t$,nom$
5050 READ t$,items
```

```
5060 READ t$:FOR i=0 TO items-1:READ nom
$(i):NEXT i
5070 RESTORE 4070
5080 somme=0:READ t$:FOR i=0 TO items-1:
READ t:somme=somme+t:NEXT i
5090 RESTORE 4070
5100 READ t$:FOR i=0 TO items-2:READ t:a
(i+1)=(t/somme)*360+a(i):NEXT i
5110 RETURN
```

Commentaires

Lignes 5080-5100 : Les valeurs des données à tracer sont d'abord additionnées pour connaître le total représenté par le cercle. Puis le pointeur DATA est ramené (RESTORE) au début des nombres de quantités. Chaque quantité est traduite en un deuxième nombre qui, lorsqu'il est divisé par 360, donne le même résultat que la division du total par la quantité initiale. Ainsi, un total 100 et une quantité d'une donnée de 25 donnerait 90, soit 25 % de 360. Ces nouvelles valeurs permettront ultérieurement de déterminer la portion du cercle à attribuer à chaque donnée.

Test

Entrez les lignes suivantes de ce qui deviendra le module de contrôle, puis exécutez le programme :

```
1040 GOSUB 5000
1100 END
```

Si tout s'est bien passé, rien n'apparaît. Ce n'est qu'en cas d'erreur que vous verrez quelque chose. Toutefois, si vous voulez vous rassurer, vous pouvez imprimer le contenu des variables et des tableaux du module.

Module 2.2.3 : Préparation de l'écran

Ce module appelle le mode graphique et les couleurs associées, et trace un cercle de 80*80 au centre de l'écran, avec le nom du diagramme.

MODULE 2.2.3 : LIGNES 2000-2090

```
2000 REM ***********************
2010 REM *Traçage de la courbe*
2020 REM ***********************
```

```
2030 MODE 1:ORIGIN 320,184:BORDER 0
2040 INK 0,0:INK 1,2:INK 2,6:INK 3,18
2050 LOCATE 20-LEN(nom$)/2,1:PEN 2:PRINT
 nom$
2060 DEG
2070 PLOT 0,184,3
2080 FOR a=0 TO 360 STEP 15:DRAW 184*SIN
(a),184*COS(a):NEXT a
2090 RETURN
```

Commentaires

Ligne 2080 : Si vous lisez les explications concernant les calculs liés au traçage d'un cercle dans les commentaires du premier programme de ce livre (chapitre 1), vous verrez que cette ligne trace tout simplement un cercle. Plutôt que de représenter chaque point individuellement sur la circonférence, une série de points distants de 15 degrés sont calculés, puis reliés par des lignes.

Test

Ajoutez les lignes suivantes puis exécutez le programme :

1060 GOSUB 2000
1080 IF INKEY$=»» THEN GOTO 1080

Vous n'obtenez rien de plus que le titre du diagramme et un cercle jaune. Pour revenir à l'écran normal, frappez une touche autre que ESC.

Module 2.2.4 : Insertion des détails

Ce module trace les segments qui vont diviser le diagramme, les colorie et place les libellés spécifiés dans le module DATA. Pour bien comprendre ce processus, rappelez-vous de la création d'un cercle, telle que nous l'avons expliquée dans le programme Heurclassique. N'hésitez pas à revenir sur ce programme et à en relire les commentaires.

MODULE 2.2.4 : LIGNES 3000-3170

```
3000 REM ************************
3010 REM *Insertion des segments*
3020 REM ************************
3030 FOR i=0 TO items-1
3040 MOVE 0,0:DRAW 184*SIN(a(i)),184*COS
```

```
(a(i))
3050 NEXT i
3055 PEN 1
3060 FOR i=0 TO items-1
3070 ta=((a(i)+a(i+1+items*(i+1=items)))
)/2
3080 IF a(i+1+items*(i+1=items))<a(i) TH
EN ta=ta+180
3090 MOVE 180*SIN(ta),180*COS(ta):c=i MO
D 3+1:IF i=items-1 AND c=1 THEN c=2
3100 GOSUB 6000
3110 tx=184*SIN(ta)
3120 ty=184*COS(ta)
3130 dx=320*SGN(tx)
3140 MOVE tx,ty:DRAW dx,ty,3
3150 LOCATE 1+(LEN(nom$(i))-40)*(dx=320)
,14-INT((ty+9)/16):PRINT nom$(i);
3160 NEXT i
3170 RETURN
```

Commentaires

Lignes 3030-3050 : Plusieurs lignes sont tracées du centre du cercle vers la circonférence, divisant le cercle en segments. Les nombres utilisés sont ceux que nous avons calculés au module 2.2.2.

Lignes 3070-3100 : D'autres angles sont calculés, mais cette fois-ci ils sont situés au milieu de chacun des segments que nous venons de tracer. L'expression logique des lignes 3070 et 3080 (DONNEES*(I+1=DONNEES) fait que, lorsque le dernier segment a été atteint, le début du premier segment sert de deuxième angle pour calculer la position médiane. Pour ce dernier calcul, la réponse sera fausse : au lieu d'additionner un angle avec un angle supérieur et de diviser par 2 pour trouver le point médian, l'angle du dernier segment est ajouté à 0 (le début du premier segment), ce qui donne un angle situé à mi-distance entre le début du premier et du dernier segment. Cette erreur est rectifiée en ajoutant 180 degrés à la dernière réponse.

Ensuite, la ligne 3090 calcule une position sur la base de cet angle juste à l'intérieur de la circonférence du cercle. Ensuite, un appel du module suivant, très semblable au module de «remplissage» utilisé avec Heurclassique, permet de colorer le segment en forme de secteur où se trouve le point en cours. La

ligne 3090 produit également un cycle régulier des trois couleurs d'avant-plan spécifiées dans le module d'initialisation, à l'exception du dernier segment qui ne peut pas avoir la même couleur que le premier, pour faciliter la lecture.

Lignes 3110-3120 : Ces deux lignes calculent un point sur la circonférence du cercle, à un angle situé à mi-distance entre le point de départ et d'arrivée du segment en cours.

Ligne 3130 : Lorsque nous avons créé le cadre du graphique, nous avons déjà déplacé l'origine des commandes graphiques au centre de l'écran. Cette ligne calcule une position sur l'écran à 320 pixels du centre. La variable TX contient déjà la coordonnée X d'un point sur la circonférence qui peut être soit négatif (à gauche du centre), soit positif (à droite du centre). En multipliant 320 par SGN(TX), si le centre du segment est à gauche du centre de l'écran, il en sera de même pour DX, et vice-versa.

Lignes 3140-3150 : Une ligne est tracée de la circonférence du cercle vers le bord gauche ou droit de l'écran, d'après la définition de DX. A l'extrémité de la ligne, ou plutôt sur elle, est placé le libellé du segment sur lequel pointe la ligne. La position d'affichage des libellés à droite est déplacée à gauche, pour qu'ils ne débordent pas des limites de l'écran ; pour cela, une condition logique permet de déterminer la coordonnée X. La ligne paraît plus complexe qu'elle ne l'est en réalité, en raison de la nécessité de traduire la position calculée précédemment en pixels, en une position de caractères.

Test

Ajoutez les lignes suivantes et exécutez le programme :

```
1070 GOSUB 3000
6000 RETURN
```

Vous devez obtenir un affichage comme celui du début de la section de ce programme, avec des segments individuels non coloriés.

Module 2.2.5 : Remplissage des segments

Ce module est pratiquement identique à celui du programme Heurclassique, à la seule différence que le remplissage se poursuit jusqu'à la rencontre d'une couleur différente de la couleur de fond.

MODULE 2.2.5 : LIGNES 6000-6300

```
6000 REM *************
6010 REM *Remplissage*
6020 REM *************
6030 IF TESTR(0,0)<>0 THEN RETURN
6040 s=2
6050 MOVE s*INT(XPOS/s),2*INT(YPOS/2)
6060 :
6070 :
6080 REM recherche haut/droite ********
6090 IF TESTR(0,s)=0 THEN GOTO 6090
6100 IF TESTR(s,-s)=0 THEN GOTO 6090
6110 :
6120 :
6130 REM recherche haut/gauche ********
6140 MOVER -s,0
6150 IF TESTR(0,s)=0 THEN GOTO 6090
6160 IF TESTR(-s,-s)=0 THEN GOTO 6150
6170 :
6180 :
6190 REM Coloriage des lignes *********
6200 x1=XPOS
6210 MOVER s,0
6220 PLOTR 0,0,c
6230 IF TESTR(s,0)=0 THEN GOTO 6220
6240 x2=XPOS-s
6250 MOVE x1,YPOS-s
6260 IF TESTR(s,0)=0 THEN GOTO 6290
6270 IF XPOS=x2 THEN RETURN
6280 GOTO 6260
6290 IF TESTR(-s,0)<>0 THEN GOTO 6200
6300 GOTO 6290
```

Test

Identique au module précédent, mais à présent les segments sont coloriés.

Module 2.2.6 : Le module de contrôle

La plupart des lignes de ce module ont déjà été entrées, mais assurez-vous, avec le listage ci-après, que rien ne manque.

MODULE 2.2.6 : LIGNES 1000-1120

```
1000 REM **********
1010 REM *Controle*
1020 REM **********
1030 ON ERROR GOTO 1110
1040 GOSUB 5000
1050 ON BREAK GOSUB 1090
1060 GOSUB 2000
1070 GOSUB 3000
1080 IF INKEY$="" THEN GOTO 1080
1090 CLEAR:CLS:LIST 4000-4999
1100 END
1110 PRINT"  *** FORMAT DES DONNEES PROB
ABLEMENT INCORRECT ***"
1120 FOR i=1 TO 3000:NEXT i:GOTO 1090
```

Commentaires

Lignes 1030 et 1110-1120 : Bref message d'erreur indiquant qu'il y a probablement une erreur dans le module DATA. Ce n'est pas un test infaillible, car certaines anomalies ne généreront pas d'erreurs détectables par ON ERROR.

Test

Remplacez l'un des nombres sous l'en-tête 'QUANTITES' par une lettre, puis exécutez le programme. Vous devez voir le message d'erreur suivi du module DATA. Corrigez cette erreur délibérée et exécutez à nouveau le programme. Cette fois-ci, en appuyant deux fois sur ESC (ou une seule fois sur une autre touche lorsque le diagramme est terminé), vous devez obtenir le listage du module DATA.

PROGRAMME 2.3 : HISTOGRAMME 3D

Fonction du programme

Après avoir présenté des données en mode haute résolution de deux façons différentes, nous allons voir les excellents résultats obtenus sur le 464 avec le jeu graphique basse résolution. En effet, en utilisant les caractères graphiques basse résolution qui n'existent pas sur le clavier mais que vous trouverez à l'annexe 3 du manuel, vous pouvez obtenir des effets 'prêts à l'emploi' qui seraient très difficiles à réussir en haute résolution.

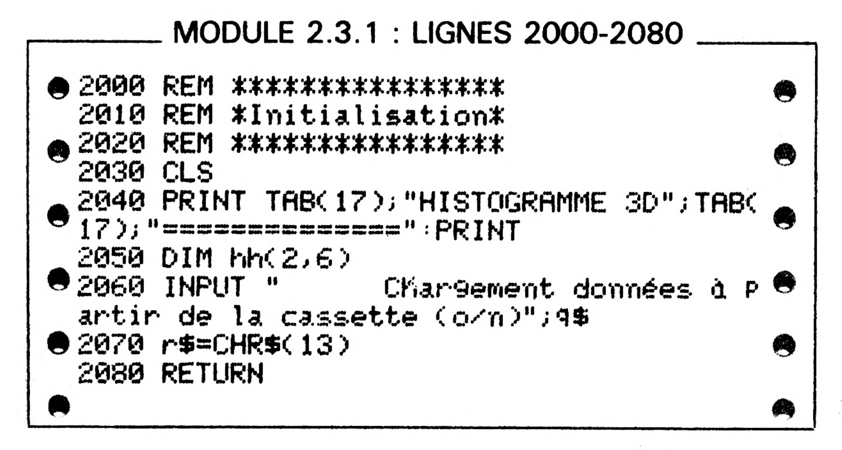
Avec le programme suivant, nous allons créer un histogramme à trois dimensions qui, à mon avis, démontre à merveille le côté spectaculaire de l'affichage basse résolution du 464 - le genre d'affichage devant lequel toute la famille va s'extasier.

Dans ce programme, vous allez découvrir les nouveaux concepts suivants :

1) L'utilisation du lecteur pour stocker les données du programme.
2) L'utilisation des caractères graphiques basse résolution.
3) L'entrée interactive des données.

Module 2.3.1 : Initialisation

Un module simple pour déclarer un petit tableau et pour demander si les données doivent être chargées à partir de la cassette. Nous reviendrons sur ce point ultérieurement.



MODULE 2.3.1 : LIGNES 2000-2080

```
2000 REM ****************
2010 REM *Initialisation*
2020 REM ****************
2030 CLS
2040 PRINT TAB(17);"HISTOGRAMME 3D";TAB(
17);"==============":PRINT
2050 DIM hh(2,6)
2060 INPUT "        Chargement données à p
artir de la cassette (o/n)";q$
2070 r$=CHR$(13)
2080 RETURN
```

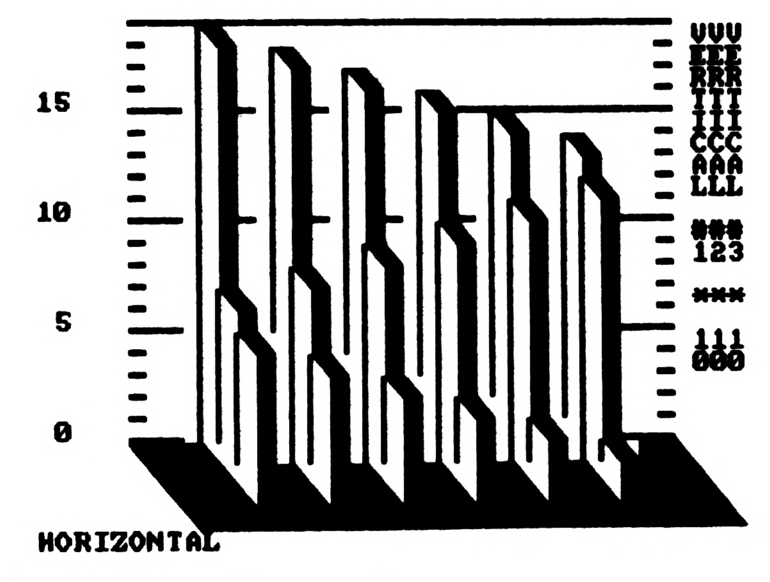


Figure 2.3 : Affichage de Histogramme 3D.

Commentaires

Ligne 2050 : Le tableau HH sera utilisé pour stocker les données de l'histogramme. Comme les nombres de l'instruction DIM doivent être comptés à partir de zéro, nous avons ici de l'espace pour trois ensembles de sept données.

Ligne 2060 : Cette entrée (INPUT) permet à un module ultérieur d'appeler un ensemble de données pour un histogramme à partir d'une cassette.

Ligne 2070 : La chaîne R$ sera utilisée dans le module de fichier de données et nous fournirons les explications à ce moment-là.

Module 2.3.2 : Acceptation des données

Dans ce dernier programme du chapitre, nous n'allons pas utiliser l'artifice des instructions DATA pour stocker les informations variables. La plupart des programmes pratiques permettent à l'utilisateur d'entrer ce type d'informations pendant l'exécution du programme. On dit que ces programmes sont 'interactifs'. Dans le cas présent, toutes les informations peu-

vent être rassemblées en une fois ; donc, le module demande les informations, les utilise pour en demander d'autres, et s'assure que les entrées ne contiennent pas d'erreurs.

MODULE 2.3.2 : LIGNES 3000-3250

```
3000 REM *************************
3010 REM *Acceptation des données*
3020 REM *************************
3030 CLS
3040 PRINT TAB(17);"HISTOGRAMME 3D";TAB(
17);"==============":PRINT
3050 PRINT"Il y a 19 unités verticalemen
t.":PRINT
3060 INPUT"Nombre représenté par chaque
unité:",uv:PRINT
3070 INPUT"Colonnes (1-6):",nd:PRINT
3080 INPUT"Blocs (1-3):",nb:PRINT
3090 PRINT"*****************************
***********":PRINT
3100 INPUT "Nom de l'axe des abscisses:"
,nh$:PRINT
3110 FOR i=0 TO nb-1
3120 PRINT"Nom de l'axe des ordonnées ";
i+1;:INPUT nv$(i):PRINT
3130 NEXT i
3140 CLS
3150 FOR i=0 TO nb-1
3160 FOR j=1 TO nd
3170 t=20*uv
3180 WHILE t/uv>19
3190 PRINT:PRINT"Entrée du bloc";i+1;"va
leur";j;":";
3200 INPUT t
3210 IF INT(t/uv)>19 THEN PRINT:PRINT"**
**** Valeur trop grande ******"
3220 WEND
3230 hh(i,j)=t
3240 NEXT j,i
3250 RETURN
```

Commentaires

Lignes 3050-3060 : Comme avec le programme de courbe précédent, chaque unité sur l'axe vertical peut représenter toute valeur spécifiée par l'utilisateur. A noter que, travaillant en

basse résolution, nous n'avons pas la même souplesse que précédemment quant à la taille des unités verticales. La seule taille pratique pour chaque unité est la hauteur du carré d'un caractère, et le format de l'histogramme permet 19 unités sur l'axe vertical.

Lignes 3070-3080 : Comme nous l'avons dit, l'histogramme accepte trois ensembles de six données. L'affichage se fera sous la forme de trois lignes pouvant contenir jusqu'à six barres pleines. Chaque ligne de barres sera appelée un 'bloc' et chacune des barres sera appelée 'colonne'.

Lignes 3110-3130 : Comme nous pouvons avoir jusqu'à trois blocs, il nous faut aussi jusqu'à trois noms pour l'axe des abscisses, par exemple prix de vente, coût, bénéfice. A noter que les noms doivent être placés dans le tableau NV$ (Nom de Verticale) à partir de l'élément 0. Ce tableau n'est pas dimensionné dans la routine d'initialisation car il n'aura toujours que trois éléments. La simple mention du nom du tableau dans le programme définit automatiquement le tableau avec dix éléments (0-9).

Lignes 3150-3240 : Vous êtes invité à fournir, pour chaque bloc, les données correspondant au nombre de colonnes spécifié. Le programme s'assure que ce nombre n'amène pas le graphique au-delà de la marque de la 19ème unité. La boucle des lignes 3180-3220 s'assure qu'aucun nombre entré ne nécessite un histogramme supérieur à 19 carrés de caractères sur l'écran. Si vous entrez un tel nombre, le programme vous alerte et vous en demande un autre.

Test

Entrez les lignes suivantes (certaines feront partie du module de contrôle du programme) :

```
1030 GOSUB 2000
1040 GOSUB 3000
1110 CLS :END
```

Puis exécutez le programme que vous venez d'entrer et répondez de la façon suivante à ses questions :

Pour le nombre représenté par chaque unité verticale : 1
Colonnes : 1
Blocs : 1
Nom de l'axe des abscisses : HORIZONTAL
Nom de l'axe des ordonnées 1 : VERTICAL 1
Bloc 1, valeur 1 : 10

Après l'entrée du dernier nombre, l'écran s'efface et le programme s'arrête avec le message 'READY'. Entrez :

? UV,ND,NB,NH$,NV$(0),HH(0,1)

Vous devez obtenir :

1	1	1
HORIZONTAL	VERTICAL 1	10

Si vous le voulez, vous pouvez exécuter à nouveau le programme en entrant des valeurs fausses. Vous ne devez pas pouvoir entrer dans une colonne, une valeur supérieure à 19 fois la valeur d'une seule unité de l'axe des ordonnées.

Module 2.3.3 : Traçage du cadre de l'histogramme

Comme pour la courbe, il faut tracer un cadre pour que l'information présentée prenne tout son sens. C'est le rôle du présent module, qui utilise des boucles, des variables et la commande LOCATE pour placer des caractères basse résolution sur les positions correctes de l'écran.

MODULE 2.3.3 : LIGNES 4000-4260

```
4000 REM ******************
4010 REM *Traçage du cadre*
4020 REM ******************
4030 CLS
4040 FOR i=0 TO 3
4050 LOCATE 6+i,21+i
4060 PEN 2:PRINT CHR$(213);STRING$(29,CH
R$(143));CHR$(215)
4070 NEXT i:PEN 1
4080 LOCATE 6,1:PRINT STRING$(30,CHR$(21
0))
4090 FOR i=2 TO 20
4100 LOCATE 6,i:PRINT CHR$(210)
4110 LOCATE 35,i:PRINT CHR$(210)
4120 NEXT i
4130 FOR i=5 TO 20 STEP 5
4140 LOCATE 6,i:PRINT STRING$(30,CHR$(21
0))
4150 NEXT i
4160 LOCATE 1,25:PRINT nh$
4170 FOR h=0 TO nb-1
4180 PEN h+1
```

```
4190 tt$=nv$(h)+" *"+STR$(uv)
4200 FOR i=1 TO LEN (tt$)
4210 LOCATE 37+h,i+1:PRINT MID$(tt$,i,1)
4220 NEXT i,h
4230 FOR i=0 TO 15 STEP 5
4240 LOCATE 1,20-i:PRINT USING "##";i
4250 NEXT i
4260 RETURN
```

Commentaires

Lignes 4040-4070 : Cette boucle affiche la base de l'histogramme. Vous trouverez la liste des caractères graphiques à l'annexe 3 du manuel.

Ligne 4080 : Une ligne en haut de l'écran, constituée du caractère numéro 210 (une barre horizontale), répétée 30 fois.

Lignes 4090-4120 : Les lignes verticales de marques placées à chaque extrémité de l'histogramme représentent les 19 unités possibles.

Lignes 4130-4150 : Quatre lignes au travers de l'histogramme, représentant des unités de cinq sur l'axe des ordonnées.

Ligne 4160 : Libellé de l'axe des abscisses.

Lignes 4170-4220 : Ces boucles affichent le(s) nom(s) de l'axe des ordonnées sur le bord droit de l'écran, avec la valeur représentée par chaque unité. La valeur I de la boucle permet de faire la navette entre les noms des axes et de changer la couleur de chaque nom. La couleur de chaque titre correspondra à la couleur de l'un des blocs.

Lignes 4230-4250 : Cette boucle place des nombres sur les cinq marques d'unités à gauche de l'histogramme.

Test

Entrez une autre ligne :

1050 GOSUB 4000

puis exécutez le programme. Spécifiez la valeur de l'unité, une colonne et trois blocs. Donnez les trois noms de l'axe des ordonnées. Lorsque le programme vous demande de spécifier la valeur des colonnes, il vous suffit d'appuyer sur RETURN. Le cadre de l'histogramme doit s'afficher, avec le nom de l'axe des abscisses en bas, et le nom de l'axe des ordonnées en haut et à droite.

Module 2.3.4 : Traçage de l'histogramme

Ce module sera bien plus facile à comprendre lorsque vous l'aurez entré et que vous aurez vu l'effet produit. Il ne repose pas tant sur des principes généraux que sur les résultats d'expériences pour voir quelles combinaisons de caractères dans certaines positions fournissent l'effet le plus satisfaisant.

MODULE 2.3.4 : LIGNES 5000-5220

```
5000 REM *******************
5010 REM *Traçage des blocs*
5020 REM *******************
5030 FOR h=0 TO nb-1
5040 PEN h+1
5050 FOR i=nd TO 1 STEP -1
5060 condition=1:WHILE INT(hh(h,i)/uv)<>
0 AND condition
5070 FOR j=1 TO INT(hh(h,i)/uv)+1
5080 LOCATE 9+4*(i-1)+h,22+h-j
5090 IF j=1 THEN PEN 2:PAPER 0:PRINT CHR
$(143);CHR$(215);:PEN h+1:PRINT CHR$(143
)
5100 IF j>1 AND j<INT(hh(h,i)/uv)+1 THEN
 PRINT CHR$(209);" ";CHR$(143)
5110 IF j=INT(hh(h,i)/uv)+1 THEN PRINT C
HR$(209);CHR$(213);CHR$(215)
5120 NEXT j
5130 condition=0:WEND
5140 NEXT i
5150 NEXT h
5160 FOR i=0 TO nd-1
5170 FOR j=1 TO nb
5180 LOCATE 9+4*i+j,20+j
5190 IF j>1 OR hh(2,i)=0 OR (j=1 AND nd=
0) THEN PEN 2:PRINT CHR$(215)
5200 NEXT j
5210 NEXT i
5220 RETURN
```

Commentaires

Lignes 5030-5150 : Cette boucle crée le nombre de blocs spécifiés.

Lignes 5050-5140 : La deuxième boucle crée le nombre de colonnes spécifiées, de droite à gauche.

Lignes 5060-5130 : Ces lignes créent une boucle WHILE fictive pour éviter la section qui trace une colonne si la valeur de cette colonne est 0.

Cette boucle est fictive car, quelles que soient les circonstances, elle n'est jamais exécutée plus d'une fois. Pour cela, la valeur de la variable CONDITION devient l'une des conditions de répétition de la boucle. Le fait de donner la valeur 0 à CONDITION avant d'atteindre WEND à la ligne 5130, garantit que la boucle n'est jamais exécutée plus d'une fois. Prenez bonne note de cette technique, car le 464 ne dispose pas d'une commande EXIT permettant de mettre fin automatiquement à la boucle.

Lignes 5070-5120 : Cette boucle crée une colonne. La ligne 5080 donne la position de tracé de chaque caractère dans la colonne. La position commence à l'extrême droite du nombre de colonnes (représenté par I), remonte la colonne individuelle (indiquée par J) et se déplace de quatre espaces vers la gauche pour chaque nouvelle colonne. En outre, lorsqu'un bloc est terminé, le bloc de colonnes suivant (représenté par H) s'affiche un espace plus bas et à droite du dernier.

La variable J de la boucle est, comme nous l'avons dit, prévue pour aller de 1 à la hauteur de la colonne. Lors du premier passage dans la boucle, la base de la colonne est créée (ligne 5090). Lors des passages suivants, différents caractères représentent le côté et l'avant de la colonne pendant son élaboration, l'instruction LOCATE servant au positionnement. Enfin, le sommet de la colonne est ajouté (ligne 5110).

Lignes 5160-5210 : Lorsque les colonnes sont terminées, les bords inférieurs ne sont pas très réguliers, et ces lignes les rendent plus nets.

Test

Entrez une nouvelle ligne :

```
1060 GOSUB 5000
```

et exécutez le programme. Spécifiez une valeur d'unité de 1, trois colonnes et trois blocs. Les noms des axes ne sont pas importants, aussi vous pouvez choisir selon vos goûts. En réponse aux demandes de valeurs de colonnes, entrez :

6,12,18,6,12,18,6,12,18

Les trois blocs et les trois colonnes doivent s'afficher clairement, le sommet de chacune des trois colonnes présentant une surface unie du bloc arrière à l'avant. A noter, que lors de la lecture des valeurs des trois blocs, vous devez supposer que le sommet de la barre la plus avant continue vers l'arrière et vers le haut, vers la position la plus arrière, la valeur de la colonne étant lue à partir du bord arrière de la colonne tel qu'il apparaîtrait dans le bloc le plus arrière. Dans l'exemple affiché, vous avez trois colonnes avec les trois barres de chaque colonne représentant la même valeur, bien que la barre avant soit physiquement plus basse sur l'écran. Cela est nécessaire pour donner l'illusion des trois dimensions.

Faites des essais avec ce programme, pour voir les résultats obtenus avec des valeurs différentes. Vous constaterez qu'il ne fonctionne correctement qu'à la condition que les données d'un bloc ne soient jamais supérieures à celles du bloc situé derrière. Ce type de représentation répond à de nombreux cas pratiques, par exemple les données de prix de vente/coût/bénéfice, citées précédemment.

Modules 2.3.5 et 2.3.6 : Stockage des données sur cassette

Nous abordons pour la première fois un sujet que de nombreuses personnes négligent, à leurs dépens, dans leurs propres programmes : le stockage des données sur cassette. Vous avez certainement constaté que, si vous avez fait une ou deux erreurs en entrant le programme, le fait de recommencer l'entrée des données devient rapidement irritant. En outre, pour de nombreux programmes, à quoi sert-il d'entrer les données dans l'ordinateur si vous devez vous les rappeler chaque fois que vous éteignez le 464 ? Peut-être avec le programme actuel, n'est-il pas totalement irréaliste d'entrer à nouveau les données, mais songez à des programmes plus complexes avec des centaines de données !

Vous avez donc deviné l'intérêt d'un magnétophone à cassette permettant d'enregistrer de façon permanente les données entrées, puis de les relire dans le 464 quand elles sont nécessaires. Les deux modules suivants accomplissent cette tâche.

MODULE 2.3.5 et 2.3.6 : LIGNES 6000-7110

```
6000 REM *************************
6010 REM *Sauvegarde des données*
6015 REM *     sur cassette      *
```

```
6020 REM ************************
6030 OPENOUT "Donnéesgraph"
6040 PRINT #9,nb;r$;nd;r$;nh$;r$;uv
6050 FOR i=0 TO nb-1
6060 PRINT #9,nv$(i)
6070 FOR j=0 TO nd
6080 PRINT #9,hh(i,j)
6090 NEXT j,i
6100 PRINT #9:CLOSEOUT
6110 RETURN
7000 REM *************************
7010 REM *Chargement des données à*
7015 REM *  partir de la cassette *
7020 REM *************************
7030 OPENIN "Donnéesgraph"
7040 INPUT #9,nb,nd,nh$,uv
7050 FOR i=0 TO nb-1
7060 INPUT #9,nv$(i)
7070 FOR j=0 TO nd
7080 INPUT #9,hh(i,j)
7090 NEXT j,i
7100 CLOSEIN
7110 RETURN
```

Commentaires

Ligne 6030 : Avant de stocker des données sur cassette, vous devez leur réserver une place, lors d'une opération appelée 'ouverture d'un fichier'. Voici le format utilisé :

OPENOUT «NOMFICHIER»

Lignes 6040-6090 : Après avoir ouvert (OPEN) le fichier, il ne reste plus qu'à y enregistrer des informations. Pour cela, nous utilisons une commande spéciale PRINT #. Les données transférées dans le fichier comprennent le contenu du tableau principal (HH$), plus les variables principales.

Ligne 6040 : Au sujet de l'instruction PRINT #, il faut noter la présence d'un certain nombre de R$ dans la ligne. Peut-être vous souvenez-vous que, dans le premier module du programme, R$ était égal à CHR$(13), soit le caractère RETURN qui signifie la fin d'une donnée à imprimer. Lorsque vous transférez plusieurs données dans un fichier avec une seule instruction PRINT #, toutes les données seront traitées ensemble, sauf si elles sont séparées par R$.

Ligne 6060 : Nous venons d'indiquer que R$ (ou une autre variable égale à CHR$(13)), devait être inclus pour séparer les données. Aussi, pourquoi ne faisons-nous pas de même pour ces deux boucles qui transfèrent le contenu de deux tableaux dans le fichier sur cassette ? Tout simplement parce que chaque fois qu'une instruction PRINT ou PRINT # se termine sans ponctuation, le 464 fait suivre automatiquement la dernière donnée transférée d'un caractère RETURN. C'est pourquoi les données sont affichées sur des lignes séparées de l'écran, si la donnée précédente n'est pas terminée par une virgule ou un point-virgule.

Ligne 6100 : Il est judicieux de terminer le transfert dans un fichier par un simple PRINT # <NUMERO FICHIER⩽ vide, qui garantit que toute donnée en attente d'affichage dans la mémoire du 464 est effacée. Enfin il faut fermer (CLOSE) le fichier. L'instruction CLOSEOUT est utilisée pour un fichier dans lequel des données ont été sauvegardées. Si vous ne faites pas cela, le numéro de fichier ne pourra pas être utilisé ultérieurement et vous risquez même de perdre les données sur cassette lorsque le programme essaie une relecture.

Lignes 7000-7110 : Ces lignes jouent le rôle inverse des lignes précédentes : elles lisent des données ultérieurement stockées sur cassette.

Ligne 7030 : Là aussi, il faut ouvrir (OPEN) un fichier. Seule différence, nous utilisons ici la forme OPENIN qui indique au système que nous voulons prendre les données *à partir de* la cassette.

Lignes 7040-7090 : INPUT # est l'opposé de PRINT #. A noter qu'il n'est pas nécessaire d'utiliser le séparateur R$ lors de l'entrée (INPUT) de données. Il est inhérent aux instructions INPUT et INPUT A de ne pas reconnaître la réception d'une donnée tant que la touche ENTER n'est pas actionnée ou qu'un caractère RETURN n'est pas lu sur la cassette.

Ligne 7100 : Comme à la ligne 6100, *il faut* fermer le fichier lorsque vous en avez fini avec lui.

Test

Pour tester ce module, il vaut mieux attendre d'avoir entré les quelques lignes qui terminent le module de contrôle du programme.

Module 2.3.7 : Le module de contrôle

Vous avez déjà entré bon nombre des lignes de ce module dans les procédures de test du programme. Il ne reste plus qu'à vérifier que le module est terminé en pointant avec le listage suivant.

MODULE 2.3.7 : LIGNES 1000-1110

```
1000 REM **********
1010 REM *Controle*
1020 REM **********
1030 GOSUB 2000
1040 IF LOWER$(q$)="o" THEN GOSUB 7000 E
LSE GOSUB 3000
1050 GOSUB 4000
1060 GOSUB 5000
1070 WHILE INKEY$=""
1080 WEND
1090 LOCATE 1,25:INPUT "   Sauvegarder l
es données sur cassette (o/n)";q$
1100 IF LOWER$(q$)="y" THEN GOSUB 6000
1110 CLS:END
```

Test

Exécutez simplement le programme. A présent, vous pouvez saisir les données d'un histogramme. Dès que l'histogramme a été affiché, le fait d'actionner une touche affiche le message de sauvegarde des données. Avant de répondre 'Y' (Oui), vérifiez qu'une cassette appropriée se trouve bien dans le lecteur (afin de ne pas écraser un programme existant). Lorsque le programme a terminé la sauvegarde des données, exécutez-le à nouveau et cette fois-ci, répondez par 'Y' (Oui) lorsqu'il vous demande si vous voulez charger à partir de la cassette. Le même histogramme doit apparaître. Si le test est satisfaisant, le programme est opérationnel.

3

Son et Lumière

Dans ce chapitre, nous étudierons voir de plus près les possibilités du 464 dans le domaine sonore et graphique. Le plus souvent, vous incorporerez à vos programmes quelques petites routines graphiques ou sonores, même si ce n'est rien de plus que le module de titre fort simple du dernier chapitre ou le signal sonore à deux tons. Parfois, il s'agira de quelque chose de plus ambitieux : un dessin complexe ou une mélodie pour animer un programme. Dans ces cas, plutôt que d'écrire chaque fois une routine séparée de création du dessin ou de la mélodie, il vaut mieux disposer de quelques outils permettant de créer plus facilement des modèles que vous pourrez lire sur cassette pour des programmes ultérieurs.

Dans ce chapitre, nous vous proposons trois de ces outils qui vous permettront de créer des dessins en haute résolution, de créer vos propres jeux de caractères et d'écrire de la musique sur votre 464. Partant du contenu de ces programmes, vous ne serez limité dans l'utilisation du son et du graphisme dans vos propres programmes, que par votre imagination.

Voici les programmes de ce chapitre :

CARACTERES : Permet de créer des jeux de caractères personnalisés.

ARTISTE : Un outil pour créer des dessins haute résolution.

MUSIQUE : Permet d'écrire des mélodies en deux parties sous une forme simple, puis de les rejouer.

PROGRAMME 3.1 : CARACTERES

Fonction du programme

Après avoir vu au chapitre précédent ce qui est possible avec le graphisme haute résolution, nous allons voir l'autre aspect, le graphisme basse résolution, avec un programme vous permettant de modifier la forme des caractères affichés par le 464.

Mais avant de passer au programme proprement dit, voici quelques mots sur le principe de création et d'affichage des caractères en mode basse résolution.

L'écran basse résolution du 464 accepte 25 lignes de 40 caractères, soit 1000 espaces de caractères. Autrement dit, vous pourriez afficher 1 000 éléments séparés sur l'écran, bien que certains soient les mêmes, car le 464 ne peut bien entendu pas générer 1 000 caractères *différents* en même temps. Toutefois cette limite de 1 000 n'est que virtuelle. Si vous observez de près n'importe quel caractère affiché, vous constatez qu'il n'est pas composé de traits pleins, comme le texte que vous êtes en train de lire, mais de points. En fait, chacun des 1 000 espaces de caractères de l'écran est constitué de 64 positions de points et c'est la combinaison de ces 64 points qui représente chaque caractère affichable par le 464. Ainsi, la figure 3.1 présente la structure de la lettre 'A'.

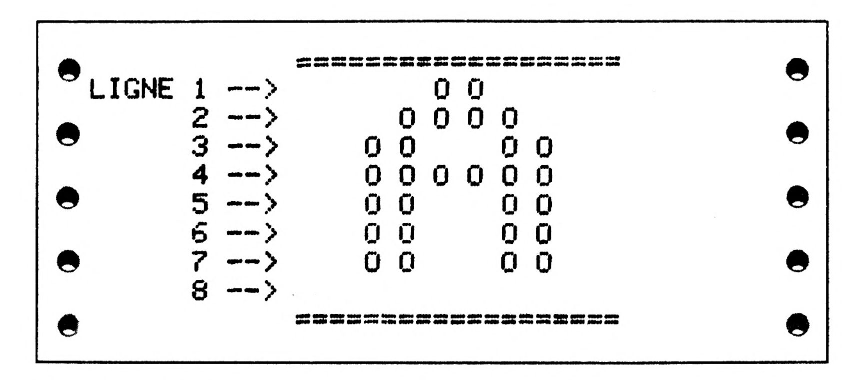


Figure 3.1 : Agrandissement de l'affichage de la lettre 'A'.

Les points qui constituent les caractères sont appelés 'pixels',

abréviation de 'picture elements' (éléments de dessin); ils représentent le plus petit élément que le 464 peut manipuler sur l'écran, en modes haute ou basse résolution.

Ces formes complexes ne sont pas dûes au hasard. Il est évident que, quelque part dans la mémoire du 464, quelque chose fait que, lorsque vous frappez la touche 'A', la combinaison de points de l'illustration apparaît sur l'écran. En fait, tous les caractères que le 464 peut afficher sont stockés sous forme de chiffres, dans un bloc de mémoire appelé la 'mémoire de caractères' ou 'ROM de caractères' (ROM = Read Only Memory = Mémoire morte). Chaque caractère dispose de huit octets de mémoire, et chacun de ces octets détermine l'endroit où les pixels apparaîtront sur une ligne du caractère. Pour cela, la valeur de chaque octet représente une ligne en système de numération binaire utilisée par le 464. Les nombres sont exprimés en puissances de 2, et non de 10, comme dans notre système de comptage normal (décimal).
Ainsi :

2013006

dans notre façon de compter habituelle représente :

(2*10^6)+(0*10^5)+(1*10^4)+(3*10^3)+(0*10^2)+(0*10^1)
+(6*10^0)

Mais, en binaire, les seuls chiffres permis sont '1' et '0' et un nombre tel que :

11001010

représente :

(2^7)+(2^6)+(2^3)+(2^1) en ignorant les zéros.

Il n'est pas nécessaire de maîtriser la numération binaire; à condition de savoir qu'un seul octet de mémoire du 464 peut contenir un nombre binaire de huit chiffres, et que tous ces uns et ces zéros représentent parfaitement quels pixels sont allumés dans une ligne de caractères. Par exemple, la lettre 'A' correspond aux huit nombres binaires suivants :

0 0 0 1 1 0 0 0
0 0 1 1 1 1 0 0
0 1 1 0 0 1 1 0
0 1 1 0 0 1 1 0
0 1 1 1 1 1 1 0
0 1 1 0 0 1 1 0
0 1 1 0 0 1 1 0
0 0 0 0 0 0 0 0

En y regardant de plus près, vous pouvez encore distinguer la lettre 'A', représentée par des uns au lieu de points, alors qu'il serait bien plus difficile de la reconnaître sous la forme :

24,60,102,102,126,102,102,0

qui est le résultat de la traduction des nombres binaires dans notre bon vieux système décimal.

Tout cela revient à dire que, lorsqu'il doit afficher un caractère, le 464 explore sa 'mémoire de caractères' et utilise ce qu'il y trouve pour dessiner le caractère sur l'écran.

Il en découle que, s'il est possible de modifier le contenu de la mémoire de caractères, cela influence directement la forme des caractères affichés. Nous pouvons donc personnaliser des lettres, des caractères graphiques, en somme tout ce qui rentre dans le carré de caractères élémentaire de 8*8.

Malheureusement, la mémoire de caractères *ne peut pas* être modifiée. Elle fait partie de la mémoire morte (ROM = Read Only Memory) intégrée dans le 464. Par contre, nous *pouvons* la copier ailleurs dans la mémoire vive (RAM), où nous pourrons la modifier, puis utiliser les nouvelles données avec la puissante commande SYMBOL pour modifier les caractères eux-mêmes. Ainsi, nous pouvons manipuler le jeu de caractères à notre gré. C'est le but du programme suivant.

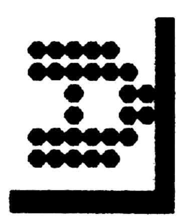


Figure 3.2 : Affichage de Histogramme 3D.

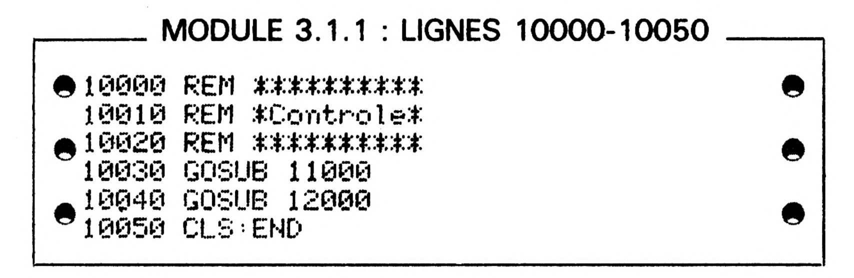
L'affichage présente l'écran pendant le mode d'édition de caractères, avec une lettre 'A' qui a subi une rotation de 90 degrés.

```
Numéro de caractère : 65

'i' pour inverser
'm' pour mirer
'r' pour retourner
'1' pour remplir le carré
'0' pour vider le carré
't' pour tourner
'p' pour placer en mémoire
's' pour sauvegarder sur cassette
'l' pour charger à partir cassette
'x' pour normaliser le jeu de caractères
'e' pour terminer
Touches curseur pour déplacer
```

Module 3.1.1 : Le module de contrôle

En général, nous ne présentons pas un module de contrôle avant l'opération d'entrée d'un programme, mais celui-ci est court et, en son absence, le programme lui-même nécessiterait un GOSUB pour être lancé.



MODULE 3.1.1 : LIGNES 10000-10050

```
10000 REM **********
10010 REM *Controle*
10020 REM **********
10030 GOSUB 11000
10040 GOSUB 12000
10050 CLS:END
```

Module 3.1.2 : Création du nouveau jeu de caractères

Ce module accomplit la tâche présentée dans l'introduction du programme : le transfert des données de caractères dans une partie de la mémoire où nous pourrons les manipuler. Il s'agit

en fait d'un tableau appelé C%, qui nous donne toute la souplesse nécessaire pour stocker et changer les données.

MODULE 3.1.2 : LIGNES 11000-11190

```
11000 REM ****************
11010 REM *Initialisation*
11020 REM ****************
11030 CLS:PEN 1:PRINT "Premier caractère
 défini par l'utilisateur":INPUT "(32-25
5)";fc
11040 IF fc<32 OR fc>255 THEN PRINT " **
**HORS LIMITES:RECOMMENCEZ ****":GOTO 11
030
11050 SYMBOL AFTER fc
11060 IF in=0 THEN ch=fc:DIM a%(7),t%(7)
,c%(255,7):in=1
11070 menu$="imr10tpslxe"+CHR$(240)+CHR$
(241)+CHR$(242)+CHR$(243)
11080 MODE 2
11082 LOCATE 36,14:PRINT "ATTENDEZ"
11084 LOCATE 1,1
11090 FOR i=fc TO 255
11100 PRINT CHR$(i);
11110 NEXT i
11120 FOR i=0 TO 255-fc
11130 FOR j=0 TO 7
11140 c%(fc+i,j)=PEEK(48*1024+j*2048+i)
11150 NEXT j
11160 NEXT i
11170 MODE 1
11180 a$="r"
11190 RETURN
```

Commentaires

Lignes 11030-11050 : Le 464 vous permet de définir tout caractère dans la limite de 16 à 255. Mais, vous ne pouvez en définir plus de 16 que si vous en manifestez l'intention par la commande SYMBOL AFTER. Cette commande suivie d'un nombre signifie qu'à partir de ce moment, vous voulez pouvoir modifier tout caractère à partir de celui qui est représenté par le nombre suivant SYMBOL AFTER, jusqu'au caractère numéro 255, avec la commande SYMBOL que nous allons voir dans un instant. Ces lignes vous permettent d'établir le premier

caractère en mesure d'être redéfini, entre le caractère numéro 32 et 255. Normalement, le 464 vous permet de redéfinir 16 caractères, entre le numéro 240 et le numéro 255.

Ligne 11060 : A l'exception d'un cas spécifique d'utilisation du programme, cette ligne permet d'établir le tableau principal, C%, qui contiendra les données de caractères (256 lignes avec huit nombres par ligne) et deux tableaux utilisés pour la manipulation temporaire d'un caractère individuel.

Ligne 11070 : Cette ligne est importante mais nous expliquerons sa signification lorsque nous arriverons au module qui traite du menu de programme.

Lignes 11080-11170 : Une belle astuce. Plutôt que de prendre les données en mémoire, il est bien plus simple d'aller les chercher sur un endroit beaucoup plus évident : l'écran lui-même. L'avantage est clair : dans le futur, si vous voulez obtenir les données de caractères sélectionnés, il vous suffit d'afficher ces caractères sur l'écran. Pour cela, nous passons d'abord en MODE 2, le mode haute résolution. La raison étant qu'en MODE 2, l'agencement des pixels sur l'écran correspond exactement au modèle décrit dans l'introduction du programme. En MODE 1, les chiffres binaires individuels ne sont pas reproduits exactement par les pixels sur l'écran, car *deux* pixels sont affichés horizontalement pour chaque chiffre binaire.

Le mode étant changé, le travail suivant consiste simplement à afficher tous les caractères à partir de FC (le premier caractère) (First Character) spécifié par l'utilisateur, jusqu'à 255, de la première position de l'écran, vers l'avant. La deuxième paire de boucles (lignes 11120-11160) lit ensuite les octets de mémoire qui donnent l'affichage sur l'écran. Cette opération n'est pas aussi directe qu'elle pourrait, en raison de l'agencement de la mémoire d'écran. Le bloc de mémoire qui reçoit le contenu de l'écran est agencé de telle sorte (nous n'en donnons pas les raisons ici) que, si vous voulez trouver les huit octets qui constituent le caractère de l'angle supérieur gauche (en MODE 2), vous devez examiner l'octet de mémoire 49152 pour l'octet supérieur du caractère, puis l'octet 51200 (49152+2048) pour le suivant vers le bas, et ainsi de suite par intervalles de 2048 octets pour les huit octets. Le caractère suivant à droite commence à l'octet 49153 et progresse par intervalles de 2048 octets. Lorsque nous avons atteint la première ligne de caractères, un déplacement d'un octet vers l'avant nous amène au premier octet du premier caractère de la deuxième ligne. Si tout cela ne vous paraît pas encore très clair, essayez d'entrer le petit programme de test suivant :

```
10 MODE 2
20 FOR I=49152 TO 49152+1024
30 POKE I,255
40 NEXT I
```

Vous verrez que, bien que vous manipuliez (POKE) la mémoire d'écran par intervalles d'un octet, la ligne obtenue par l'allumage des pixels, longe la partie supérieure des lignes de caractères.

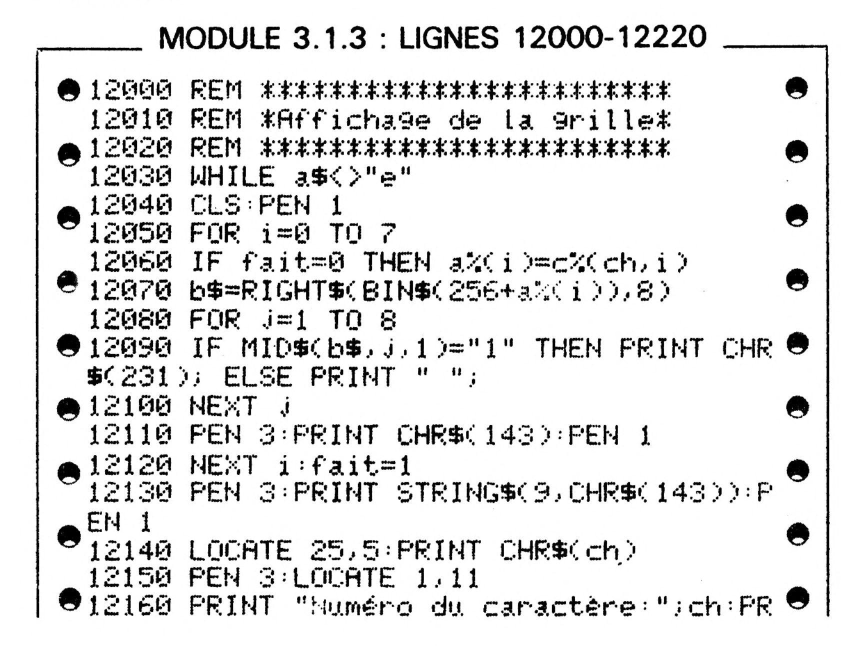
Cela étant, vous pouvez voir ce qu'accomplit la ligne 11140, qui lit un à un les octets des données de caractères à partir de la mémoire d'écran.

Ligne 11180 : Cette ligne définit une variable utilisée pour stocker les réponses du menu, afin que le programme ne se termine pas accidentellement, à la première exécution.

Module 3.1.3 : Affichage d'un caractère agrandi

Ce programme a pour but de faciliter l'édition (modification) de caractères. Une méthode consiste à afficher une version agrandie d'un caractère spécifié afin de pouvoir y déplacer un curseur. Ce module permet de spécifier puis d'afficher le caractère.

MODULE 3.1.3 : LIGNES 12000-12220



```
12000 REM ************************
12010 REM *Affichage de la grille*
12020 REM ************************
12030 WHILE a$<>"e"
12040 CLS:PEN 1
12050 FOR i=0 TO 7
12060 IF fait=0 THEN a%(i)=c%(ch,i)
12070 b$=RIGHT$(BIN$(256+a%(i)),8)
12080 FOR j=1 TO 8
12090 IF MID$(b$,j,1)="1" THEN PRINT CHR
$(231); ELSE PRINT " ";
12100 NEXT j
12110 PEN 3:PRINT CHR$(143):PEN 1
12120 NEXT i:fait=1
12130 PEN 3:PRINT STRING$(9,CHR$(143)):P
EN 1
12140 LOCATE 25,5:PRINT CHR$(ch)
12150 PEN 3:LOCATE 1,11
12160 PRINT "Numéro du caractère:";ch:PR
```

```
INT
12170 IF a$="r" THEN INPUT "Numéro de dé
placement pointeur (0 pour redéfinir) "
;mm:ch=ch+mm
12180 IF ch<fc THEN ch=fc
12190 IF ch>255 THEN ch=255
12200 IF mm=0 THEN GOSUB 13000 ELSE fait
=0
12210 WEND
12220 RETURN
```

Commentaires

Lignes 12040-12120 : Les données d'un caractère particulier sont lues en A%, à partir du tableau C%, puis utilisées pour afficher une version agrandie du caractère. La méthode pour cela consiste à utiliser d'abord BIN$ pour traduire chaque octet des données de caractères en une suite de 1 et de 0. Puis un caractère cercle plein (CHR$(231)) est affiché pour représenter la position d'un 1 dans la suite. Le motif 8*8 est entouré d'une bordure pleine constituée de CHR$(143). Pour finir, la variable DONE est définie à la fin, de sorte que, si le module est répété sans modifications du caractère, les données ne sont pas recopiées en A%.

Ligne 12140 : Une version en taille normale du caractère est placée à droite de la version agrandie.

Lignes 12170-12190 : A la première exécution du programme, le caractère affiché sera le premier de la gamme qui peut être redéfini, telle qu'elle figure dans la variable FC. Ces lignes permettent de déplacer le caractère visé, à l'intérieur des limites FC et 255.

Test

Exécutez le programme et spécifiez 65 comme premier caractère défini par l'utilisateur. Après l'affichage de la gamme des caractères et une pause permettant le transfert des données, une version agrandie de la lettre 'A' doit apparaître dans l'angle supérieur gauche. Vous pouvez aussi explorer le jeu de caractères et examiner tout caractère à partir de 'A'. A noter que vous ne pouvez encore rien *faire* sur le caractère affiché. Pour cela, nous frapperons '0' pour appeler un module suivant. Normalement le programme s'arrête lorsque le mode '0' est appelé, mais, dans ce module, vous pouvez faire la même chose en appuyant sur ESC.

Module 3.1.4 : Un curseur défini par l'utilisateur

Un module simple permettant d'afficher une imitation de curseur à l'endroit choisi.

MODULE 3.1.4 : LIGNES 14000-14110

```
14000 REM **********************
14010 REM *Affichage du curseur*
14020 REM **********************
14025 b$=RIGHT$(BIN$(256+a%(y)),8)
14030 LOCATE x+1,y+1:PAPER 2:PEN 1
14040 IF MID$(b$,x+1,1)="1" THEN PRINT C
HR$(231); ELSE PRINT " ";
14050 a$=""
14060 WHILE a$=""
14070 a$=LOWER$(INKEY$)
14080 WEND
14090 LOCATE x+1,y+1:PAPER 0
14100 IF MID$(b$,x+1,1)="1" THEN PRINT C
HR$(231);ELSE PRINT " ";
14110 RETURN
```

Commentaires

Lignes 14030-14040 : L'apparence du curseur est obtenue en changeant les couleurs PAPER et INK pour simuler la présence du curseur normal, puis en réaffichant soit un espace soit le petit cercle plein utilisé pour afficher les versions agrandies de caractères.

Lignes 14050-14080 : Etat d'attente jusqu'à ce qu'une touche soit actionnée.

Lignes 14090-14100 : La position de caractère où se trouve le curseur s'affiche à nouveau en couleurs normales.

Test

Entrez :

RUN 14000[RETURN]

Un carré de curseur doit apparaître dans l'angle supérieur gauche de l'écran. Frappez une touche et le programme s'arrête avec un message d'erreur 'Unexpected RETURN' (RETURN inattendu).

Module 3.1.5 : Entrée des commandes

Ce module combine les fonctions suivantes : présentation d'un menu, déplacement du curseur clignotant, coloriage ou effacement des pixels agrandis, et acceptation des commandes pour manipuler le caractère en cours. Le menu donne toutes les indications nécessaires à l'utilisation. Le seul aspect original du module réside dans la façon dont les instructions sont acceptées.

MODULE 3.1.5 : LIGNES 13000-13220

```
13000 REM ******************************
13010 REM *Redéfinition d'un caractère*
13020 REM ******************************
13030 LOCATE 1,13:PRINT STRING$(40," ")
13040 LOCATE 1,13
13050 PRINT "'i' pour inverser"
13060 PRINT "'m' pour mirer"
13070 PRINT "'r' pour retourner"
13080 PRINT "'1' pour remplir le carré"
13090 PRINT "'0' pour vider le carré"
13100 PRINT "'t' pour tourner"
13110 PRINT "'p' pour placer en mémoire"
13120 PRINT "'s' pour sauvegarder sur ca
ssette"
13130 PRINT "'l' pour charger à partir d
e la cassette"
13140 PRINT "'x' pour normaliser le jeu
de caractères"
13150 PRINT "'e' pour terminer"
13160 PRINT "Touches curseur pour déplac
er"
13170 ret=0:WHILE ret=0
13180 GOSUB 14000
13190 z=INSTR(menu$,a$)
13200 ON z GOSUB 15000,16000,17000,18000
,19000,20000,21000,22000,23000,24000,250
00,26000,27000,28000,29000
13210 WEND
13220 RETURN
```

Commentaires

Lignes 13170-13210 : Vous vous souvenez sûrement que, dans le module d'initialisation, nous avons créé une chaîne plu-

tôt étrange appelée MENU$. Cette ligne en est la raison. Ensemble, elles permettent de traduire en un nombre un simple enfoncement de touche, grâce à INSTR. La position d'un caractère dans MENU$ est la valeur qui sera donnée à Z et utilisée pour ON Z GOSUB sur la ligne suivante. La variable RET indique au module s'il est nécessaire de retracer tout l'affichage. Si, au retour de l'exécution, RET est égal à 0, le caractère agrandi n'est pas retracé, d'où un gain de temps.

Test

Exécutez le programme. Après que le 'A' ait été tracé, vous pouvez entrer 0 et voir le menu affiché. Aucune des commandes n'aura d'autre effet que d'arrêter le programme avec un message d'erreur.

Module 3.1.6 : Retour du mode édition

La touche 'R' frappée à partir du menu principal retourne l'exécution au module précédent, pour vous permettre de traiter le caractère suivant.

MODULE 3.1.6 : LIGNES 17000-17030

```
17000 REM ********
17010 REM *Retour*
17020 REM ********
17030 ret=1:RETURN
```

Test

Vous êtes à présent en mesure de passer du menu principal au module de manipulation des caractères, et inversement.

Module 3.1.7 : Déplacement du curseur

Après nous être donné un curseur, nous allons pouvoir le déplacer, tout simplement en modifiant les valeurs des variables X et Y des coordonnées du curseur.

MODULE 3.1.7 : LIGNES 26000-29040

```
26000 REM ******
26010 REM *Haut*
26020 REM ******
```

```
26030 IF y>0 THEN y=y-1
26040 RETURN
27000 REM *****
27010 REM *Bas*
27020 REM *****
27030 IF y<7 THEN y=y+1
27040 RETURN
28000 REM ********
28010 REM *Gauche*
28020 REM ********
28030 IF x>0 THEN x=x-1
28040 RETURN
29000 REM ********
29010 REM *Droite*
29020 REM ********
29030 IF x<7 THEN x=x+1
29040 RETURN
```

Test

Le menu principal étant affiché, vous pouvez à présent déplacer le curseur sur le caractère agrandi, bien que vous ne puissiez pas encore toucher au dessin.

Module 3.1.8 : Coloriage et effacement

En entrant '1' ou '0' à partir du menu principal, vous pouvez colorier ou effacer un pixel individuel sur le dessin agrandi. Dans ce module, nous utilisons pour cela les opérateurs logiques AND et OR.

MODULE 3.1.8 : LIGNES 18000-19060

```
18000 REM *********************
18010 REM *Coloriage du carré*
18020 REM *********************
18030 a%(y)=a%(y) OR 2^(7-x)
18040 LOCATE x+1,y+1
18050 PRINT CHR$(231)
18060 RETURN
19000 REM **********************
19010 REM *Effacement du carré*
19020 REM **********************
19030 a%(y)=a%(y) OR 255-2^(7-x)
```

```
19040 LOCATE x+1,y+1
19050 PRINT " "
19060 RETURN
```

Commentaires

Ligne 18030 : L'utilisation de OR est une méthode courante pour donner la valeur 1 à un ou plusieurs chiffres binaires d'un nombre. OR compare deux nombres binaires, puis en produit un troisième où chaque chiffre binaire reçoit la valeur 1 si le chiffre binaire *de l'un des* nombres d'origine était 1. Par exemple, avec les deux nombres d'origine :

124 = 01111100

et

3 = 00000011

le résultat serait 127, ou 01111111, puisque chaque chiffre binaire avait la valeur 1 dans l'un ou l'autre des deux nombres.

Si nous voulons imposer la valeur 1 à un chiffre binaire particulier, numéro N en lisant de droite à gauche, il nous suffit d'utiliser OR 2N.

Dans le cas du programme, le fait de donner la valeur 1 à un chiffre binaire particulier revient à allumer un pixel, puisque l'état de ces derniers est déterminé par l'état des chiffres binaires des huit nombre qui constituent le caractère.

Ligne 19030 : L'opposé de OR, pour ce qui touche à l'activation (valeur 1) et la désactivation (valeur 0) des chiffres binaires individuels, est AND. Lorsque deux nombres sont reliés par AND, les chiffres du nombre résultant n'ont la valeur 1 que dans les positions où *les deux* nombres d'origine avaient cette même valeur. Ainsi, si nous relions 124 et 3 par AND comme ci-dessus, nous obtenons 0, puisqu'aucun des chiffres binaires n'est activé dans les deux nombres. Le fait de relier par AND un nombre entre 0 et 255 et 255 ne présente aucune différence, puisque dans le nombre 255, chacun des huit chiffres prend la valeur 1. Toutefois, chacun des chiffres de 255 peut être désactivé (valeur 0) en soustrayant 2N de 255, où N est le numéro du chiffre qui doit recevoir la valeur 0. Par conséquent, si vous reliez par AND un nombre, X, avec 255-2N, le chiffre binaire N prend la valeur 0 dans X.

Du point de vue du programme, cela équivaut à effacer un pixel.

Test

En mode d'édition de caractères, vous pouvez déplacer le curseur et colorier (remplir) ou effacer (vider) des pixels à volonté.

Module 3.1.9 : Création d'un caractère inversé

Nous passons à présent à une suite de trois modules qui facilitent l'édition d'un caractère, en opérant sur la totalité du caractère; ainsi, vous pouvez le refléter (mirer) ou lui faire subir une rotation de 90 degrés. Le module en cours crée simplement l'inverse du caractère existant. Tout pixel activé sera effacé, et toute position vierge recevra un pixel. Cela est obtenu par l'inversion des 1 et 0 binaires des octets qui constituent le caractère.

MODULE 3.1.9 : LIGNES 15000-15060

```
15000 REM ***********
15010 REM *Inversion*
15020 REM ***********
15030 FOR i=0 TO 7
15040 a%(i)=255-a%(i)
15050 NEXT i
15060 ret=1:RETURN
```

Test

Exécutez le programme et appelez le mode édition. Lorsque le menu est affiché, frappez 'I' pour voir un caractère inverse créé. Puis frappez 'I' à nouveau pour ramener le caractère à son état initial. A noter que cela ne concerne que le caractère agrandi et non pas celui de taille normale à droite du carré de 8*8. Le caractère de taille normale ne changera que si vous décidez d'entrer votre caractère édité en mémoire, dans un module ultérieur.

Module 3.1.10 : Création d'une réflexion (effet de miroir)

Ce module prend le caractère affiché et le présente comme s'il était reflété dans un miroir.

MODULE 3.1.10 : LIGNES 16000-16100

```
16000 REM ********
16010 REM *Miroir*
16020 REM ********
16030 FOR i=0 TO 7
16040 b$=RIGHT$(BIN$(256+a%(i)),8)
16050 a%(i)=0
16060 FOR j=0 TO 7
16070 a%(i)=a%(i)+2^j*VAL(MID$(b$,j+1,1)
)
16080 NEXT j
16090 NEXT i
16100 ret=1:RETURN
```

Commentaires

Lignes 16060-16080 : Dès que chaque octet du caractère est extrait de A% et transformé en une chaîne binaire, il est lu de gauche à droite comme une chaîne. Donc, lorsque la variable J de la boucle a la valeur 0, le chiffre lu représente 128. Donc, si le premier caractère de la chaîne binaire est un 1, il est transformé en 20, soit 1 en termes décimaux. Si le dernier caractère de la chaîne est un 1, il est traduit en 27, soit 128 en termes décimaux. Ainsi, le nombre binaire est retourné 'd'avant en arrière'.

Test

Exécutez le programme et appelez le menu d'édition. Frappez 'M' et, après un court instant pendant lequel le caractère est copié dans le tableau, vous devez le voir en réflexion.

Module 3.1.11 : Rotation d'un caractère

Imaginez que le caractère que vous éditez est imprimé sur une feuille de plastique transparent. A partir de là, à l'exception du maintien dans un angle particulier, tout ce qui est possible avec la feuille plastique peut l'être par une combinaison de réflexion et/ou de rotation du caractère de 90 degrés, une ou plusieurs fois. Le module en cours utilise à nouveau le tableau T% mais cette fois-ci pour faire tourner le caractère dans le carré 8*8, de 90 degrés vers la gauche.

MODULE 3.1.11 : LIGNES 20000-20130

```
20000 REM **********
20010 REM *Rotation*
20020 REM **********
20030 FOR i=0 TO 7
20040 b$=RIGHT$(BIN$(256+a%(i)),8)
20050 FOR j=0 TO 7
20060 IF i=0 THEN t%(j)=0
20070 t%(j)=t%(i)+2^i*VAL(MID$(b$,j+1,1)
)
20080 NEXT j
20090 NEXT i
20100 FOR i=0 TO 7
20110 a%(i)=t%(i)
20120 NEXT i
20130 ret=1:RETURN
```

Commentaires

Lignes 20050-20090 : Cette boucle copie les chiffres binaires de l'un des huit nombres de A%, dans le tableau temporaire T%. A noter toutefois que, alors que les chiffres binaires de chaque nombre de A% sont lus de gauche à droite, ils sont copiés dans T% de haut en bas : le premier chiffre binaire de chaque nombre de A% va dans le premier élément de T%, le deuxième chiffre binaire de chaque nombre de A% va dans le deuxième élément de T%, etc. De sorte que, la colonne de gauche de chiffres binaires du caractère d'origine devient la ligne supérieure (horizontale) de T%, d'où une rotation de 90 degrés du caractère.

Lignes 20100-20120 : Le caractère réorienté est relu de T% dans A%, la zone de travail principale.

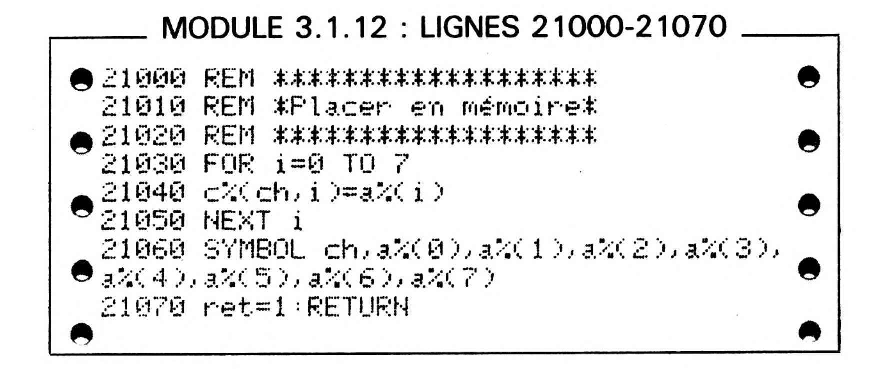
Test

Exécutez le programme, appelez le menu d'édition et frappez 'T'. Après un bref instant, le caractère est réaffiché et tourné de 90 degrés. Si vous frappez 'T' trois fois encore, le caractère doit retrouver sa position initiale. Faites quelques essais en combinant l'effet de miroir, la rotation et l'inversion, jusqu'à ce que vous soyez familiarisé avec ces techniques.

Module 3.1.12 : Entrée d'un caractère édité en mémoire

Jusqu'à présent, vous avez pu manipuler le caractère agrandi dans le carré 8*8, peut-être jusqu'à ce qu'il n'ait plus aucun point commun avec sa présentation de départ. Tout cela n'a rien changé à la version de taille normale du caractère affiché à droite du carré 8*8. Les modifications apportées jusqu'ici n'ont pas encore été transférées dans la mémoire des caractères et elles ne le seront que lorsque votre création vous conviendra. Dès que vous *avez* ce que vous souhaitiez, ce module va intégrer le motif inscrit dans le carré, dans votre jeu de caractères personnalisé.

MODULE 3.1.12 : LIGNES 21000-21070



```
21000 REM *******************
21010 REM *Placer en mémoire*
21020 REM *******************
21030 FOR i=0 TO 7
21040 c%(ch,i)=a%(i)
21050 NEXT i
21060 SYMBOL ch,a%(0),a%(1),a%(2),a%(3),
a%(4),a%(5),a%(6),a%(7)
21070 ret=1:RETURN
```

Commentaires

Ligne 21040 : Le contenu de A% (le tableau dans lequel toutes les manipulations de données de caractères ont lieu) est copié dans la position de C% correspondant au caractère en cours.

Ligne 21060 : Les données de A% sont utilisées avec la puissante commande SYMBOL pour affecter de nouvelles valeurs aux huit octets qui enregistrent la forme du caractère en cours.

Test

Exécutez le programme et déplacez le pointeur sur le caractère 65, la lettre 'A'. Passez en mode édition et faites une rotation du caractère. Puis, frappez 'P' et observez l'écran. Le 'A' à droite du carré 8*8 se trouve sur le côté, parce que le 464 prend ses informations sur les caractères dans votre jeu personnalisé. Conseil : N'éditez les caractères que si vous voulez un texte personnalisé ; sinon, contentez-vous d'éditer les caractè-

res graphiques. Si vous apportez trop de modifications aux lettres et aux chiffres, vous aurez peut-être du mal à comprendre ce que le programme essaiera de vous dire!

Module 3.1.13 : Stockage du jeu de caractères

Le jeu de caractères étant édité, nous voulons pouvoir le conserver pour le réutiliser ultérieurement, afin de ne pas avoir travaillé inutilement. Ce module stocke vos caractères personnalisés sur cassette.

MODULE 3.1.13 : LIGNES 22000-22110

```
22000 REM ************************
22010 REM *Sauvegarde sur cassette*
22020 REM ************************
22030 LOCATE 1,24:OPENOUT "Données car"
22040 PRINT #9,fc
22050 FOR i=fc TO 255
22060 FOR j=0 TO 7
22070 PRINT #9,c%(i,j)
22080 NEXT j
22090 NEXT i
22100 CLOSEOUT
22110 ret=1:RETURN
```

Commentaires

Lignes 22040-22090 : Pour gagner du temps lors du chargement et de la sauvegarde, seuls les caractères redéfinis à partir de FC sont sauvegardés. En effet, il est inutile d'attendre que les données des 256 caractères soient sauvegardées, si une vingtaine seulement ont été modifiées. Les données sont sauvegardées sous la forme des nombres stockés dans C%.

Test

Après avoir modifié quelques caractères et les avoir mis en mémoire, appelez ce module pour les sauvegarder sur cassette. La seule vérification pour l'instant consiste à s'assurer que ce transfert s'effectue sans aucune erreur. Après avoir entré le module suivant, vous pourrez recharger le jeu de caractères pour vérifier qu'il a été correctement stocké.

Module 3.1.14 : Rechargement d'un jeu de caractères

Ce module permet de recharger le jeu de caractères précédemment stocké sur cassette. Tel qu'il est, ce module peut être utilisé pour recharger un jeu de caractères provenant d'autres programmes. Après avoir reconçu votre jeu de caractères et l'avoir stocké sur cassette, il vous suffit d'inclure ce module (avec une renumérotation correcte éventuellement) dans le programme qui va utiliser les nouveaux caractères. Lisez le jeu de caractères et le voilà installé dans le nouveau programme.

MODULE 3.1.14 : LIGNES 23000-23170

```
23000 REM ************************
23010 REM *Chargement à partir de*
23015 REM *      la cassette      *
23020 REM ************************
23030 LOCATE 1,24
23040 OPENIN "Données car"
23050 INPUT #9,fc
23060 FOR i=fc TO 255
23070 FOR j=0 TO 7
23080 INPUT #9,c%(i,j)
23090 NEXT j
23100 NEXT i
23110 CLOSEIN
23120 SYMBOL AFTER fc
23130 FOR i=fc TO 255
23140 SYMBOL i,c%(i,0),c%(i,1),c%(i,2),c
%(i,3),c%(i,4),c%(i,5),c%(i,6),c%(i,7)
23150 NEXT i
23160 fait=0:ret=1
23170 RETURN
```

Commentaires

Lignes 23040-23110 : Le caractère de départ est lu sur la cassette et comme il ne correspond peut-être pas à la valeur en cours, le contenu de la cassette est relu en C%, à partir de la position FC.

Lignes 23120-23150 : Les données étant retournées en C%, SYMBOL AFTER permet de redéfinir le jeu de caractères. Nor-

malement, cela pourrait être accompli dans la boucle de chargement des données, chaque caractère étant redéfini après la lecture de ses huit octets. En pratique, en raison d'une anomalie probable de la ROM du 464, la commande SYMBOL AFTER n'est apparemment pas reconnue pendant qu'un fichier est ouvert à destination du lecteur. Nous vous conseillons de vérifier si votre propre machine présente cette anomalie.

Test

Vous pouvez à présent recharger le jeu de caractères que vous avez sauvegardé (SAVE) pendant le test du module précédent ; il vous suffit de frapper 'L' en mode édition. Avant de recharger ce que vous venez de sauvegarder, vous devez avoir réinstauré le jeu de caractères normal (non modifié), sinon il sera impossible de détecter si des caractères différents ont été chargés à partir de la cassette.

Module 3.1.15 : Restauration du jeu de caractères

Il se peut que vous considériez être allé trop loin dans vos redéfinitions de caractères et que vous souhaitiez restaurer le jeu de caractères original. Pour cela, il vous suffit de taper 'X' sur le menu principal. Le module d'initialisation reprend la main pour que le caractère de départ puisse être redéfini. Si vous voulez redonner au 464 son état par défaut, le premier caractère doit être 240.

MODULE 3.1.15 : LIGNES 24000-24040

```
24000 REM ******************
24010 REM *Normaliser le jeu*
24015 REM *  de caractères  *
24020 REM ******************
24030 GOSUB 11000
24040 fait=0:ret=1:RETURN
```

Test

Exécutez le programme et modifiez un caractère, en stockant la définition en mémoire. Puis frappez 'E' et vous constaterez que ce caractère a retrouvé son état initial.

Module 3.1.16 : Fin du programme

En frappant 'E' à partir du menu principal, vous mettez fin au programme sans restaurer le jeu de caractères d'origine.

MODULE 3.1.16 : LIGNES 25000-25030

```
25000 REM *****
25010 REM *Fin*
25020 REM *****
25030 ret=1:RETURN
29030 IF x<7 THEN x=x+1
29040 RETURN
```

Test

Modifiez un caractère, placez le caractère redéfini en mémoire, puis mettez fin au programme en frappant 'X'. Toutes les modifications apportées doivent encore faire partie du jeu de caractères.

PROGRAMME 3.2 : ARTISTE

Fonction du programme

Vous avez sûrement vu quelques affichages impressionnants créés par des logiciels de CAO (Conception Assistée par Ordinateur). D'un doigt léger, l'ingénieur ajoute des lignes et des formes à des dessins complexes, ou efface ce qui existe déjà. A sa façon, ARTISTE va tenter d'imiter ce travail. Même s'il est moins élaboré, il vous permettra de créer des dessins complexes bien plus grands qu'un simple écran, en utilisant l'écran comme une fenêtre mobile permettant d'examiner certaines parties ou de rétrécir la totalité de la zone pour pouvoir la visualiser entièrement. Vous pouvez ajouter ou supprimer à volonté des lignes, des cercles et des rectangles, et stocker l'ensemble du dessin sur cassette pour utilisation ultérieure.

Voici les nouveaux concepts introduits dans ce programme :

1) Un curseur clignotant en haute résolution, défini par l'utilisateur.
2) Stockage des dessins sur cassettes.
3) Définition et traçage de formes géométriques.

Module 3.2.1 : Initialisation

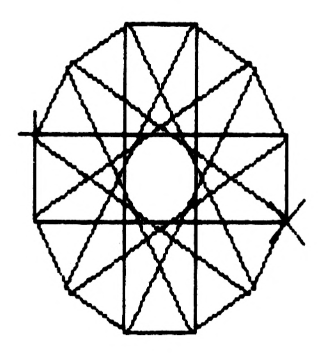


Figure 3.3 : Affichage obtenu avec Artiste.

Voici un module d'initialisation standard qui définit les couleurs de l'écran, deux fenêtres et une gamme de variables dont nous parlerons dans les commentaires du programme.

MODULE 3.2.1 : LIGNES 11000-11250

```
11000 REM ****************
11010 REM *Initialisation*
11020 REM ****************
11030 MODE 1
11040 BORDER 26
11050 INK 0,0
11060 INK 1,26
11070 INK 2,18
11080 INK 3,18
11090 PAPER 1:PEN 0
11100 CLS
11110 WINDOW 27,40,1,25
11120 ORIGIN 200,200,0,398,398,0
11130 CLG
11140 WINDOW #1,27,40,25,25:PAPER #1,1:P
EN #1,0
11150 over$=CHR$(23)+CHR$(1)
11160 normal$=CHR$(23)+CHR$(0)
11170 menu$=""
11180 FOR i=240 TO 247:menu$=menu$+CHR$(
i):NEXT i
11190 menu$=menu$+"12sclb"+CHR$(13)+"ewt
dg"
11200 unité=1
11210 pix=2
11220 gauche=-100:droite=99
```

```
11230 haut=99:bas=-100
11240 DIM a%(1000,4),x(1),y(1)
11250 RETURN
```

Module 3.2.2. : Deux curseurs de traçage

Dans tout programme de dessin, il est très important que l'utilisateur sache où se trouve la position de traçage actuelle. C'est le rôle d'un curseur de type quelconque, qui indique la position en cours sur l'écran. Ce programme utilise deux curseurs pour définir, par exemple, les deux extrémités d'une ligne tracée, les deux angles opposés d'un cadre et ainsi de suite. Les deux curseurs peuvent être déplacés sur le dessin avec les touches de contrôle du curseur, sans que le contenu de l'écran en soit affecté.

MODULE 3.3.2 : LIGNES 12000-13080

```
12000 REM ***********
12010 REM *Curseur 1*
12020 REM ***********
12030 PRINT over$
12040 PLOT x(0)*pix-16,y(0)*pix,2
12050 DRAWR 32,0
12060 PLOT x(0)*pix,y(0)*pix-16
12070 DRAWR 0,32
12080 RETURN
13000 REM ***********
13010 REM *Curseur 2*
13020 REM ***********
13030 PRINT over$
13040 PLOT x(1)*pix-16,y(1)*pix-16,2
13050 DRAWR 32,32
13060 PLOT x(1)*pix-16,y(1)*pix+16
13070 DRAWR 32,-32
13080 RETURN
```

Commentaires

Lignes 12030 et 13030 : OVER$ a été défini dans le module d'initialisation en tant que CHR$(23) plus CHR$(1). CHR$(23) est un caractère de contrôle, un caractère qui n'affi-

che rien sur l'écran mais qui influence le fonctionnement du 464.

Dans ce cas particulier, le caractère a pour effet de définir une caractéristique appelée 'OVER' de traçage sur l'écran ou, de façon plus technique, de définir un XOR (mode eXclusive OR). En clair, cela signifie que tout ce qui est tracé sur l'écran pendant que la caractéristique OVER est active, inverse l'état de tous les pixels rencontrés. S'ils étaient en couleur d'avant-plan, ils prennent la couleur de fond et inversement. L'avantage est que, si la même forme est tracée *deux fois* avec OVER actif, l'écran retrouve exactement son état précédent. Autrement dit, vous pouvez tracer un curseur puis le retracer pour l'effacer, sans que l'écran en soit affecté d'aucune façon.

Lignes 12040-12070 et 13040-13070 : Les curseurs sont tracés. L'un en forme de croix, l'autre en forme d'X. La position de la croix dans chaque case est déterminée par les tableaux X et Y à deux éléments conjointement à la valeur de la variable PIX. PIX est nécessaire car, plus loin dans le programme, vous verrez que le dessin peut être visualisé avec des échelles différentes, et qu'une distance sur le dessin peut varier lorsqu'elle est exprimée sur l'écran. PIX garantit un positionnement correct du curseur d'après l'échelle de visualisation en vigueur.

Module 3.2.3 : Sélection du curseur

Avec deux curseurs, il est bien entendu nécessaire de sélectionner celui qui doit être adressé. Le module actuel est appelé à partir du menu principal, que nous allons appeler très bientôt, à cet effet, en changeant la valeur de la variable CN (Cursor Number).

MODULE 3.2.3 : LIGNES 23000-24040

```
23000 REM *********************
23010 REM *Sélection Curseur 1*
23020 REM *********************
23030 cn=0
23040 RETURN
24000 REM *********************
24010 REM *Sélection Curseur 2*
24020 REM *********************
24030 cn=1
24040 RETURN
```

Module 3.2.4 : Déplacement des curseurs

Enfin, nous devons pouvoir déplacer les curseurs. C'est le rôle des touches de déplacement du curseur ; sans la touche shift, elles déplacent le curseur d'une unité dans la direction indiquée et, avec la touche shift, le déplacement est de 16 unités à la fois. Chacune des sous-routines du module est appelée séparément à partir du menu par sa touche appropriée. Cela allonge le programme mais accélère le déplacement du curseur par rapport à une sous-routine pleine d'instructions IF traitant les touches du curseur.

MODULE 3.2.4 : LIGNES 15000-22040

```
15000 REM ****************
15010 REM *Curseur haut 1*
15020 REM ****************
15030 y(cn)=y(cn)+unité
15040 RETURN
16000 REM ****************
16010 REM *Curseur bas 1*
16020 REM ****************
16030 y(cn)=y(cn)-unité
16040 RETURN
17000 REM ******************
17010 REM *Curseur gauche 1*
17020 REM ******************
17030 x(cn)=x(cn)-unité
17040 RETURN
18000 REM *****************
18010 REM *Curseur droit 1*
18020 REM *****************
18030 x(cn)=x(cn)+unité
18040 RETURN
19000 REM *****************
19010 REM *Curseur haut 16*
19020 REM *****************
19030 y(cn)=y(cn)+unité*16
19040 RETURN
20000 REM ****************
20010 REM *Curseur bas 16*
20020 REM ****************
20030 y(cn)=y(cn)-unité*16
20040 RETURN
21000 REM *******************
21010 REM *Curseur gauche 16*
```

```
21020 REM ******************
21030 x(cn)=x(cn)-unité*16
21040 RETURN
22000 REM ******************
22010 REM *Curseur droit 16*
22020 REM ******************
22030 x(cn)=x(cn)+unité*16
22040 RETURN
```

Module 3.2.5 : Entrée des commandes

Nous avons à présent deux curseurs et la possibilité de les déplacer. Revenons au module qui alloue le travail aux différentes parties du programe lorsqu'une touche est actionnée. Pour tester son bon fonctionnement, il faut que le module suivant, le bref module de contrôle, soit ajouté. La plupart des simples commandes citées dans le commentaire ne deviendront, bien entendu, opérationnelles qu'après l'adjonction de modules ultérieurs.

MODULE 3.2.5 : LIGNES 14000-14440

```
14000 REM **************
14010 REM *Mode Edition*
14020 REM **************
14030 CLS
14040 PRINT "Mode Edition":PRINT
14050 PRINT "X1:"
14060 PRINT "Y1:":PRINT
14070 PRINT "X2:"
14080 PRINT "Y2:":PRINT
14090 PRINT "Echelle:";unité
14100 PRINT "Items:";item:PRINT
14110 PRINT "'1' Curseur 1"
14120 PRINT "'2' Curseur 2"
14130 PRINT "'S' Forme"
14140 PRINT "'C' Cercle"
14150 PRINT "'L' Ligne"
14160 PRINT "'B' Boite"
14170 PRINT "'";CHR$(1);CHR$(13);"' Fixe
r"
14180 PRINT "'E' Effacer"
14190 PRINT "'W' Fenetre"
14200 PRINT "'T' Sauvegarde"
```

```
14210 PRINT "'D' Mode supp"
14220 PRINT "'ç' Fin"
14230 t=0:WHILE t<19
14240 GOSUB 12000
14250 GOSUB 13000
14260 LOCATE 4,3:PRINT x(0);TAB(14)
14270 LOCATE 4,4:PRINT y(0);TAB(14)
14280 LOCATE 4,6:PRINT x(1);TAB(14)
14290 LOCATE 4,7:PRINT y(1);TAB(14)
14300 t=0:WHILE t=0
14310 t$="":WHILE t$=""
14320 t$=LOWER$(INKEY$)
14330 WEND
14340 t=INSTR(menu$,t$)
14350 WEND
14360 GOSUB 12000
14370 GOSUB 13000
14380 ON t GOSUB 15000,16000,17000,18000
,19000,20000,21000,22000,23000,24000,250
00,26000,27000,28000,29000,30000,31000,3
2000
14390 IF x(cn)>droite THEN x(cn)=droite
14400 IF x(cn)<gauche THEN x(cn)=gauche
14410 IF y(cn)>haut THEN y(cn)=haut
14420 IF y(cn)<bas THEN y(cn)=bas
14430 WEND
14440 RETURN
```

Commentaires

Lignes 14030-14290 : Le menu de programme s'affiche à droite de l'écran, conformément à ce qui a été prévu lors de l'initialisation. La partie gauche et centrale de l'écran est réservée au dessin lui-même. Outre le menu, les curseurs sont tracés sur le dessin principal et leur position est indiquée.

Lignes 14300-14350 : Ces boucles imbriquées attendent qu'une touche soit actionnée; elles la comparent avec le MENU $ créé dans le module d'initialisation, produisant une valeur pour la commande ON...GOSUB de la ligne 14380.

Lignes 14390-14420 : Si, par suite d'une commande de déplacement du curseur, le curseur a dépassé les limites de la fenêtre en cours enregistrées dans les variables BAS, HAUT, GAUCHE et DROITE, les coordonnées du curseur sont modifiées par ces lignes.

Module 3.2.6 : Le module de contrôle

A ce stade, il faut entrer le bref module de contrôle pour pouvoir tester la fonction de déplacement du curseur, ainsi que les modules ultérieurs dès leur entrée. A noter que le module permet la lecture des données sur cassette ; nous reviendrons sur ce point lors de l'entrée des modules correspondants.

MODULE 3.2.6 : LIGNES 10000-10100

```
10000 REM **********
10010 REM *Controle*
10020 REM **********
10030 INPUT "Chargement à partir de la c
assette (o/n)";q$
10040 GOSUB 11000
10050 IF LOWER$(q$)="o" THEN GOSUB 33000
10060 WHILE 1
10070 GOSUB 14000
10080 IF t=20 THEN MODE 1:CLS:END
10090 GOSUB 40000
10100 WEND
```

Test

Exécutez le programme pour constater que vous pouvez déplacer chacun des curseurs avec les touches de contrôle. Aucune autre fonction du programme n'est encore disponible.

Module 3.2.7 : Traçage d'une ligne

Les trois modules suivants vont faire le travail facile et fastidieux du programme : traçage des lignes, des boîtes (carrés ou rectangles) et des polygones (y compris des cercles). Vous ne pourrez pas les utiliser pour cela immédiatement, car ces modules ne sont pas appelés directement à partir du menu, mais par d'autres modules qui ont pour tâche de gérer le traçage, l'effacement, etc. Le présent module trace simplement une ligne de la position du curseur 2 à la position du curseur 1.

MODULE 3.2.7 : LIGNES 37000-37050

```
37000 REM **********************
37010 REM *Traçage d'une ligne*
37020 REM **********************
```

```
37030 PLOT x2*Pix,y2*Pix,c
37040 DRAW x1*Pix,y1*Pix
37050 RETURN
```

Test

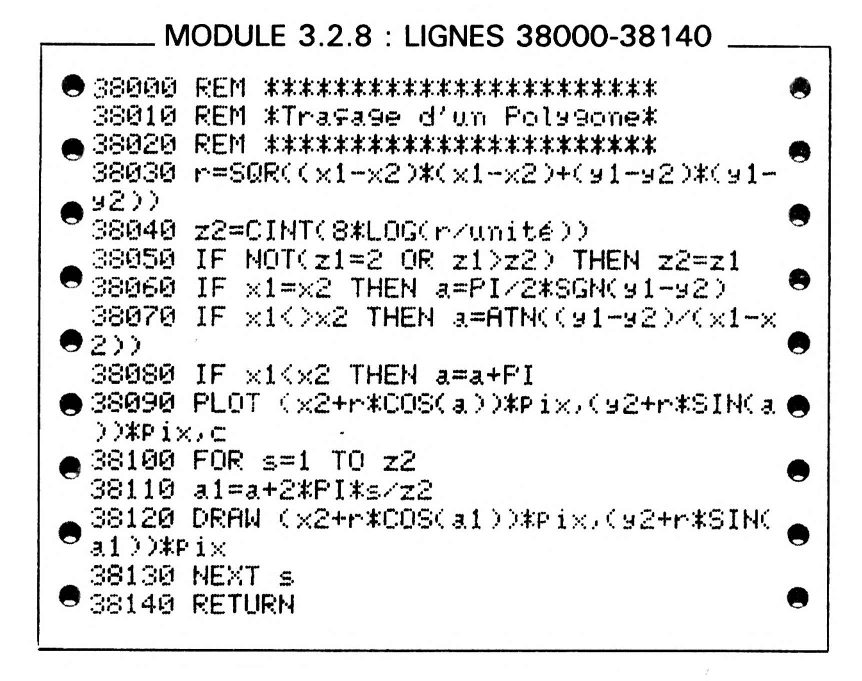
Exécutez le programme de préparation de l'écran, puis arrêtez en frappant : ' '. Tapez :

C=2 :X1=50 :Y1=50 :GOTO 37000[ENTER]

Vous devez voir une ligne tracée du centre de la zone noire, en diagonale vers le haut et vers la droite.

Module 3.2.8 : Traçage d'un cercle ou d'un polygone

Il s'agit d'un module standard qui relie des points autour d'un cercle. Selon le nombre de côtés spécifiés, vous obtiendrez un cercle ou un polygone régulier. Avec trois points seulement, la forme sera triangulaire.



MODULE 3.2.8 : LIGNES 38000-38140

```
38000 REM ************************
38010 REM *Traçage d'un Polygone*
38020 REM ************************
38030 r=SQR((x1-x2)*(x1-x2)+(y1-y2)*(y1-
y2))
38040 z2=CINT(8*LOG(r/unité))
38050 IF NOT(z1=2 OR z1>z2) THEN z2=z1
38060 IF x1=x2 THEN a=PI/2*SGN(y1-y2)
38070 IF x1<>x2 THEN a=ATN((y1-y2)/(x1-x
2))
38080 IF x1<x2 THEN a=a+PI
38090 PLOT (x2+r*COS(a))*Pix,(y2+r*SIN(a
))*Pix,c
38100 FOR s=1 TO z2
38110 a1=a+2*PI*s/z2
38120 DRAW (x2+r*COS(a1))*Pix,(y2+r*SIN(
a1))*Pix
38130 NEXT s
38140 RETURN
```

Commentaires

Ligne 38030 : Distance en diagonale entre les deux curseurs.

Lignes 38040-38050 : Z2 sera le nombre de points à placer autour du cercle. L'expression de la ligne 38040 donne suffisamment de points pour obtenir un cercle acceptable. Inutile de pousser trop loin la définition du nombre de points, car l'amélioration serait peu perceptible mais le déroulement du programme beaucoup plus lent. Si vous spécifiez une forme autre qu'un cercle, la ligne 38050 définit le nombre de points d'après celui que vous avez fourni, à condition que ce nombre ne soit pas supérieur à celui qui est nécessaire pour tracer un cercle.

Lignes 38060-38080 : Ces trois lignes calculent l'angle du curseur 1 par rapport au curseur 2.

Ligne 38090 : Premier point autour du cercle; sa position importe peu, pourvu qu'elle soit différente de la position du curseur 2. Vous reconnaissez sûrement l'équation du programme Heurclassique du premier chapitre.

Lignes 38100-38130 : Cette boucle calcule la position de Z2 points sur la circonférence du cercle, qui sont tracés parallèlement au calcul. Là encore, c'est la même formule que celle du programme Heurclassique.

Test

Compte tenu du nombre de variables nécessaires pour tester ce module et le suivant, il vaut mieux attendre que d'autres modules aient été ajoutés au programme principal.

Module 3.2.9 : Traçage d'une boîte

La dernière forme que le programme peut tracer est un simple rectangle ou carré, dont les angles opposés sont situés aux points 1 et 2. Ce module est un peu plus complexe que le précédent car il n'y a pas de commande intégrée pour cette forme et de plus nous avons prévu la rotation.

MODULE 3.2.9 : LIGNES 36000-36210

```
36000 REM **********************
36010 REM *Traçage d'une boite*
36020 REM **********************
```

```
36030 a=-z1/10000
36040 cs=COS(a)
36050 sn=SIN(a)
36060 x=(x1+x2)/2
36070 y=(y1+y2)/2
36080 rx1=x+(x1-x)*cs+(y1-y)*sn
36090 ry1=y+(y1-y)*cs-(x1-x)*sn
36100 rx2=x+(x2-x)*cs+(y1-y)*sn
36110 ry2=y+(y1-y)*cs-(x2-x)*sn
36120 rx3=x+(x2-x)*cs+(y2-y)*sn
36130 ry3=y+(y2-y)*cs-(x2-x)*sn
36140 rx4=x+(x1-x)*cs+(y2-y)*sn
36150 ry4=y+(y2-y)*cs-(x1-x)*sn
36160 PLOT rx1*Pix,ry1*Pix,c
36170 DRAW rx2*Pix,ry2*Pix
36180 DRAW rx3*Pix,ry3*Pix
36190 DRAW rx3*Pix,ry4*Pix
36200 DRAW rx1*Pix,ry1*Pix
36210 RETURN
```

Commentaires

Lignes 36030-36050 : La variable A contient l'angle, fourni par l'utilisateur, permettant la rotation de la boîte. CS et SN ont pour but de raccourcir la formule du déplacement des angles de la boîte lors de sa rotation.

Lignes 36080-36150 : Voici les formules de rotation d'un point autour d'un autre :

X2 = X0 + XD*OS ANGLE + YD*IN ANGLE
Y2 = Y0 + YD*OS ANGLE + XD*IN ANGLE

X2 et Y2 sont les coordonnées X et Y résultant de la rotation du point X,Y.

X0 et Y0 sont les coordonnées autour desquelles la rotation du point a lieu.

XD et YD sont les distances qui séparent X et Y de X0 et Y0, respectivement.

ANGLE est l'angle de rotation du point défini par X et Y.

Le rectangle de base, avant rotation, aura les angles de X1/Y1, X2/Y1, X2/Y2 et X1/Y2. Si vous examinez les lignes et si vous les comparez avec la formule précédente, vous constatez qu'elles fournissent les coordonnées après rotation des quatre points d'angle.

Module 3.2.10 : Répartition des tâches entre les modules de traçage

Ce bref module est nécessaire pour intégrer les modules de traçage avec ce qui va suivre. Nous expliquerons son utilisation en examinant le module suivant.

MODULE 3.2.10 : LIGNES 35000-35060

```
35000 REM ***************
35010 REM *Traçage global*
35020 REM ***************
35030 IF z1<1 THEN GOSUB 36000
35040 IF z1=1 THEN GOSUB 37000
35050 IF z1>1 THEN GOSUB 38000
35060 RETURN
```

Module 3.2.11 : Contrôle des commandes de traçage

Les trois sous-routines que nous allons voir à présent transforment les touches actionnées à partir du module de menu, en paramètres permettant d'utiliser les modules de traçage proprement dits.

Chacun des modules a les fonctions suivantes :

1) Si nécessaire, demander des informations complémentaires telles que 'combien de côtés' ou 'degré de rotation'.

2) Définir la caractéristique OVER en affichant OVER$.

3) Effacer toute forme existante non confirmée par l'utilisateur, comme l'indique la variable TEMP qui est égale à 1.

4) Définir les variables Z1, X1, Y1, X2 et Y2, les positions des curseurs.

5) Appeler le module de traçage approprié.

6) Définir la variable TEMP pour indiquer qu'une forme a été tracée et pas encore confirmée.

MODULE 3.2.11 : LIGNES 25000-28110

```
25000 REM *******
25010 REM *Forme*
25020 REM *******
```

```
25030 z=0:e=0:WHILE z<3 OR z>32767
25040 IF e=1 THEN x$="*HORS LIMITES*":GO
SUB 34000
25050 INPUT #1,"Cotés";z:PRINT #1
25060 e=1:WEND
25070 PRINT over$:c=2
25080 IF temp=1 THEN GOSUB 35000
25090 z1=z:x1=x(0):y1=y(0):x2=x(1):y2=y(
1)
25100 GOSUB 38000
25110 temp=1
25120 RETURN
26000 REM ********
26010 REM *Cercle*
26020 REM ********
26030 PRINT over$:c=2
26040 IF temp=1 THEN GOSUB 35000
26050 z1=2:x1=x(0):y1=y(0):x2=x(1):y2=y(
1)
26060 GOSUB 38000
26070 temp=1
26080 RETURN
27000 REM *******
27010 REM *Ligne*
27020 REM *******
27030 PRINT over$:c=2
27040 IF temp=1 THEN GOSUB 35000
27050 z1=1:x1=x(0):y1=y(0):x2=x(1):y2=y(
1)
27060 GOSUB 37000
27070 temp=1
27080 RETURN
28000 REM *******
28010 REM *Boite*
28020 REM *******
28030 INPUT #1,"Rotation";a:PRINT #1
28040 a=a-180*INT(a/180)
28050 z=-CINT(a/180*PI*10000)
28060 PRINT over$:c=2
28070 IF temp=1 THEN GOSUB 35000
28080 z1=z:x1=x(0):y1=y(0):x2=x(1):y2=y(
1)
28090 GOSUB 36000
28100 temp=1
28110 RETURN
```

Commentaires

Plutôt que de faire des commentaires sur chaque sous-routine, nous allons analyser succintement les points principaux de l'une d'entre elles. Ces points s'appliquent à toutes les sous-routines.

Lignes 25030-25060 : Information complémentaire nécessaire si une forme doit être tracée, par exemple combien de côtés.

Ligne 25080 : Si la variable TEMP est définie, une forme vient d'être tracée et n'a pas encore été confirmée (nous reviendrons sur la confirmation dans un instant). Avant de tracer la forme actuelle, le module de la ligne 35000 est utilisé pour appeler à nouveau le module de traçage approprié. Comme OVER est défini, le tracé se fera sur la forme précédente qui sera effacée. Donc, si vous voulez tracer une forme complexe qui ne convient pas tout à fait (ce n'est pas exactement ce que vous vouliez), vous pouvez déplacer l'un des curseurs, en vous servant de la forme imparfaite comme guide, puis retracer la forme en effaçant automatiquement la première version.

Ligne 25090 : Définition des variables.

Test

Vous êtes à présent capables d'exécuter le programme et tracer des formes, mais vous ne pouvez pas encore les confirmer et les intégrer dans le dessin ; en effet, chaque forme tracée effacera la précédente.

Module 3.2.12 : Enregistrement d'un dessin

Un tel programme présente peu d'intérêt tant que le dessin réalisé ne peut être enregistré. Dans notre cas, l'enregistrement du dessin dans un tableau nous permettra ultérieurement de stocker les données sur cassette et de supprimer des lignes individuelles. Les lignes et les formes sont toutes enregistrées par l'appel de ce module, au fur et à mesure de leur entrée. En outre, le module trace la forme de façon permanente afin qu'elle ne soit pas effacée lors du traçage de la suivante. Pour appeler ce module, appuyez sur ENTER sur le menu principal.

MODULE 3.2.12 : LIGNES 29000-29150

```
29000 REM *******
29010 REM *Fixer*
29020 REM *******
29030 IF temp=0 THEN RETURN
29040 IF item=1001 THEN x$="*PLUS DE PLA
CE*":GOSUB 34000:RETURN
29050 PRINT normal$:c=1
29060 GOSUB 35000
29070 a%(item,0)=z1
29080 a%(item,1)=x1
29090 a%(item,2)=y1
29100 a%(item,3)=x2
29110 a%(item,4)=y2
29120 temp=0
29130 item=item+1
29140 LOCATE 7,10:PRINT item;TAB(14)
29150 RETURN
```

Commentaires

Ligne 29030 : Si TEMP n'est pas égal à 0, il n'y a pas de forme à enregistrer.

Lignes 29050-29060 : NORMAL$, qui a été défini dans le module d'initialisation, met hors fonction OVER, de sorte que tout traçage soit normal. Puis, le module de la ligne 35000 est appelé pour permettre le choix entre les divers modules de traçage.

Lignes 29070-29130 : Les variables nécessaires au traçage de la forme sont stockées dans le tableau A%. TEMP reçoit la valeur 0 pour indiquer qu'il n'y a pas de forme non confirmée sur l'écran et la variable ITEM est incrémentée pour indiquer qu'une autre forme a été ajoutée au dessin.

Module 3.2.13 : Effacement d'une forme

Un module simple qui efface une forme non confirmée de la même façon que lorsqu'une nouvelle forme est tracée.

MODULE 3.2.13 : LIGNES 30000-30070

```
30000 REM ***********
30010 REM *Effacement*
```

```
30020 REM ***********
30030 IF temp=0 THEN RETURN
30040 PRINT over$:c=2
30050 GOSUB 35000
30060 temp=0
30070 RETURN
```

Test

Vous pouvez effacer une forme non confirmée en frappant 'E' à partir du menu.

Module 3.2.14 : Retraçage d'un dessin

Nous sommes en mesure d'enregistrer une forme dans un tableau. Voyons à présent comment retracer sur l'écran la totalité du dessin à partir des données stockées dans le tableau.

MODULE 3.2.14 : LIGNES 39000-39130

```
39000 REM ***********
39010 REM *Retraçage*
39020 REM ***********
39030 IF item=0 THEN RETURN
39040 PRINT normal$:c=1
39050 FOR i=0 TO item-1
39060 z1=a%(i,0)
39070 x1=a%(i,1)
39080 y1=a%(i,2)
39090 x2=a%(i,3)
39100 y2=a%(i,4)
39110 GOSUB 35000
39120 NEXT i
39130 RETURN
```

Test

Exécutez le programme et dessinez quelques formes, puis arrêtez le programme en frappant ' ' et tapez :

CLS #2[ENTER]
GOTO 39000[ENTER]

Toutes les formes doivent réapparaître sur l'écran.

Module 3.2.15 : Suppression des lignes et des formes

Comme le dessin n'est pas stocké de façon globale, mais sous la forme de lignes et de formes individuelles, vous pouvez facilement supprimer des données séparées. Ce module affiche tout le dessin, ligne par ligne, en vous permettant de supprimer toute ligne de votre choix. Vous pouvez aussi l'utiliser pour retracer le dessin si l'écran a été effacé par arrêt du programme.

MODULE 3.2.15 : LIGNES 40000-40290

```
40000 REM ******************
40010 REM *Mode Suppression*
40020 REM ******************
40030 CLS
40040 CLG
40050 PRINT "Mode Suppression":PRINT
40060 PRINT "Item # 1":PRINT
40070 PRINT "Echelle:";unité
40080 PRINT "Items:";item:PRINT
40090 PRINT "'";CHR$(1);CHR$(13);"' Item
 suivant"
40100 PRINT "'E' Enlever"
40110 PRINT "'ç' Retour"
40120 t$="":i=0:WHILE i<item
40130 z1=a%(i,0)
40140 x1=a%(i,1)
40150 y1=a%(i,2)
40160 x2=a%(i,3)
40170 y2=a%(i,4)
40180 d=0:WHILE t$<>"ç" AND d=0
40190 PRINT over$:c=2
40200 GOSUB 35000
40210 t$=""
40220 d=1:WEND
40230 WHILE t$=""
40240 t$=UPPER$(INKEY$)
40250 WEND
40260 IF t$="E" THEN GOSUB 41000 ELSE GO
SUB 42000
40270 WEND
40280 temp=0
40290 RETURN
```

Commentaires

Lignes 40120-40220 : Les détails d'une forme sont lus dans le tableau A% et tracés avec OVER actif.

Lignes 40230-40280 : Le programme attend que vous frappiez 'E' pour supprimer la forme affichée, ou toute autre touche pour la confirmer. Aucune des deux actions n'aura d'effet, car il reste encore à entrer les deux modules suivants.

Module 3.2.16 : Effacement d'une forme du tableau

Après avoir retracé une forme à partir du tableau, vous pouvez l'effacer définitivement du dessin tout simplement en la retraçant avec OVER actif. Ensuite, le tableau est resserré pour supprimer les détails concernant cette forme.

MODULE 3.2.16 : LIGNES 41000-41120

```
41000 REM ************
41010 REM *Effacement*
41020 REM ************
41030 PRINT over$:c=2
41040 GOSUB 35000
41050 item=item-1
41060 FOR j=i TO item-1
41070 FOR k=0 TO 4
41080 a%(j,k)=a%(j+1,k)
41090 NEXT k
41100 NEXT j
41110 LOCATE 7,6:PRINT item;TAB(14)
41120 RETURN
```

Module 3.2.17 : Confirmation d'une donnée en mode suppression

Pour confirmer une donnée, il suffit de la retracer avec OVER hors fonction.

MODULE 3.2.17 : LIGNES 42000-42070

```
42000 REM **************
42010 REM *Item suivant*
42020 REM **************
```

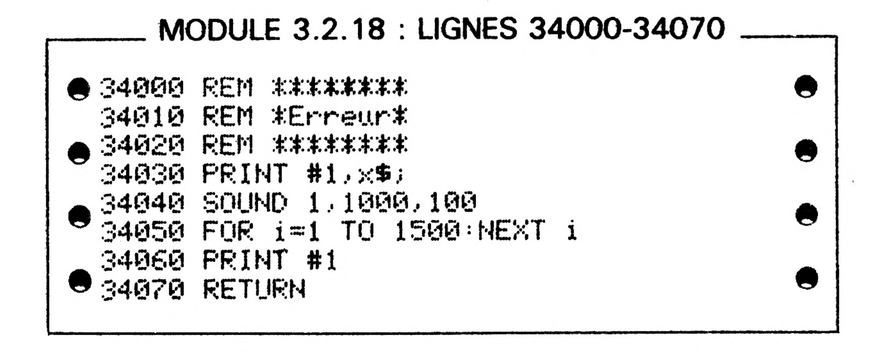
```
42030 PRINT normal$:c=1
42040 GOSUB 35000
42050 i=i+1
42060 LOCATE 7,3:PRINT i+1
42070 RETURN
```

Test

Si vous avez entré quelques formes, vous pouvez à présent obtenir ce module, en frappant 'D' à partir du module principal. Assurez-vous que vous pouvez consulter le dessin entré, supprimer des données ou les laisser intactes. Si vous frappez ' ' au début du dessin, il ne doit pas y avoir de modification, mais le dessin doit être retracé en totalité.

Module 3.2.18 : Messages d'erreur

Ce simple module affiche un message d'erreur provenant d'une autre partie du programme.



MODULE 3.2.18 : LIGNES 34000-34070

```
34000 REM ********
34010 REM *Erreur*
34020 REM ********
34030 PRINT #1,x$;
34040 SOUND 1,1000,100
34050 FOR i=1 TO 1500:NEXT i
34060 PRINT #1
34070 RETURN
```

Module 3.2.19 : Fenêtres et échelles

Nous n'avons pas encore abordé l'une des possibilités les plus intéressantes du programme ARTISTE : déplacer l'écran comme une fenêtre sur un grand dessin ou, à l'inverse, réduire un dessin pour qu'il puisse apparaître sur une seule image. C'est un module plutôt complexe, mais qui le serait bien plus sans les capacités du 464. En effet, ce dernier permet de déplacer l'origine (ORIGIN) du graphique et de tracer des lignes qui n'apparaissent pas sur l'écran, sans créer d'erreur.

MODULE 3.2.19 : LIGNES 31000-31280

```
31000 REM *********
31010 REM *Fenetre*
31020 REM *********
31030 x$="*HORS LIMITES*"
31040 e=0:WHILE e=0 OR x<-16384 OR x>163
83
31050 IF e=1 THEN GOSUB 34000
31060 INPUT #1,"X";x
31070 e=1:WEND
31080 e=0:WHILE e=0 OR y<-16384 OR y>163
83
31090 IF e=1 THEN GOSUB 34000
31100 INPUT #1,"Y";y
31110 e=1:WEND
31120 e=0:WHILE e=0 OR unité<1
31130 IF e=1 THEN GOSUB 34000
31140 INPUT #1,"Echelle";unité:PRINT #1
31150 e=1:WEND
31160 pix=2/unité
31170 LOCATE 7,9:PRINT unité;TAB(14)
31180 gauche=x-100*unité:IF gauche<-1638
4 THEN gauche=-16384
31190 droite=x+99*unité:IF droite>16383
 THEN droite=16383
31200 haut=y+99*unité:IF haut>16383 THEN
 haut=16383
31210 bas=y-100*unité:IF bas<-16384 THEN
 bas=-16384
31220 x(0)=x:y(0)=y
31230 x(1)=x:y(1)=y
31240 temp=0
31250 ORIGIN 200-x*pix,200-y*pix,0,398,3
98,0
31260 CLG
31270 GOSUB 39000
31280 RETURN
```

Commentaires

Lignes 31030-31050 : Appel de la routine de messages d'erreur que nous venons d'entrer. Le module contient plusieurs appels qui vous alertent si vous entrez des nombres incorrects.

Lignes 31060-31110 : Le programme vous demande d'entrer les coordonnées X et Y correspondant au centre de la nouvelle

fenêtre. Le dessin a une taille totale de 32768*32768 unités et la fenêtre peut se déplacer en tout point de cette zone théorique.

Lignes 31120-31150 : Entrée de l'échelle de la fenêtre. Plus ce nombre est grand, plus le dessin affiché sera petit. Ainsi, vous pouvez créer un dessin avec une échelle bien plus grande que l'écran, puis le visualiser en une seule fois grâce à une échelle réduite.

Lignes 31180-31210 : Les variables représentant les limites de l'écran sont définies. Si l'une des limites dépasse la taille maximale du dessin, la fenêtre se déplace pour la ramener dans les limites permises.

Lignes 31220-31270 : Les deux curseurs sont déplacés au centre de la nouvelle fenêtre et l'origine du graphique est redéfinie. L'écran graphique est effacé et le module de retraçage est appelé pour recréer le dessin. Il est possible que, la fenêtre étant déplacée, tout ou partie du dessin se trouve à l'extérieur de la nouvelle fenêtre et soit donc invisible.

Test

Après avoir entré un dessin, vous pouvez déplacer l'écran sur le dessin, l'agrandir ou le réduire.

Module 3.2.20 : Stockage sur cassette et relecture

Il s'agit d'une section de stockage de données standard. Pour une explication des méthodes, voir le programme Histogramme 3D au chapitre précédent.

MODULE 3.2.20 : LIGNES 32000-33130

```
32000 REM **************************
32010 REM *Sauvegarde sur cassette*
32020 REM **************************
32030 INPUT #1,"Nom:",n$:PRINT #1
32040 OPENOUT "!"+n$
32050 PRINT #9,item
32060 FOR i=0 TO item-1
32070 FOR j=0 TO 4
32080 PRINT #9,a%(i,j)
32090 NEXT j
32100 NEXT i
32110 CLOSEOUT
```

```
32120 RETURN
33000 REM ************************
33010 REM *Chargement à partir de*
33015 REM *     la cassette      *
33020 REM ************************
33030 INPUT #1,"Nom:",n$:PRINT #1
33040 OPENIN "!"+n$
33050 INPUT #9,item
33060 FOR i=0 TO item-1
33070 FOR j=0 TO 4
33080 INPUT #9,a%(i,j)
33090 NEXT j
33100 NEXT i
33110 CLOSEIN
33120 GOSUB 39000
33130 RETURN
```

Test

Créez un dessin simple et frappez 'T' pour le stocker. Arrêtez le programme et relancez-le pour effacer la mémoire. Répondez par 'Y' (Oui) au message vous demandant si vous voulez charger à partir de la cassette (TAPE). Donnez le nom du fichier sous lequel le dessin a été stocké et le dessin doit réapparaître sur l'écran.

Si ce test est concluant, le programme est opérationnel.

PROGRAMME 3.3 : MUSIQUE

Fonction du programme

Outre ses nombreux autres avantages, le 464 est sans aucun doute l'un des micros actuels les plus puissants et les plus riches sur le plan musical. De plus, contrairement à d'autres machines qui ont également beaucoup de capacités, le BASIC du 464 permet, avec un effort minimum, de tirer le meilleur parti des possibilités musicales du matériel.

Ce programme est un bon exemple de ce que vous pouvez obtenir avec un peu de réflexion. Avec lui, vous pourrez créer des mélodies et des harmonies complexes et les manipuler de diverses façons. Que vous soyez musicien confirmé ou non, les résultats permis par le programme ne manqueront pas de vous surprendre agréablement.

```
                CHANGEMENT DES PARAMETRES
                =========================

ENTER pour ne pas changer les paramètres

Octave relatif (0) :1
Note relative (0)      :3
Longueur d'unité (20) :15
Canal 0 (O)            :
Canal 1 (O)            :
Canal 2 (O)            :N
```

Figure 3.4 : Un menu du programme Musique.

Module 3.3.1 : Les données pour la mélodie

Comme avec les programmes graphiques précédents, nous allons placer l'information de base de la mélodie dans des instructions DATA en fin de programme, pour faciliter l'examen et l'édition. Il n'est pas important pour l'instant que vous compreniez toutes les données, car nous les expliquerons au fur et à mesure de l'étude du programme qui les utilise. Avec les données ci-après, vous jouerez une version très honorable de 'Fur Elise'.

MODULE 3.3.1 : LIGNES 26000-28390

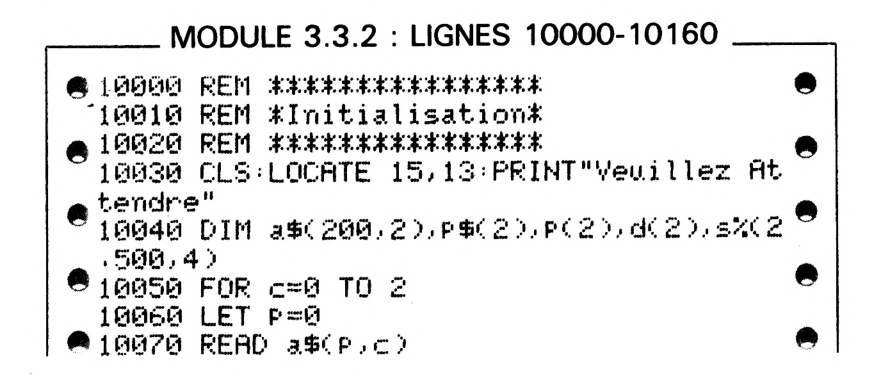
```
26000 REM *****************
26010 REM *Canal 0 Données*
26020 REM *****************
26030 DATA ENV1
26040 DATA *
26050 DATA L1,16,15
26060 DATA 16,15,16,11,14,12
26070 DATA L2,9,L1,R,0,4,9
26080 DATA L2,11,L1,R,4,8,11
26090 DATA L2,12,L1,R,4,16,15
26100 DATA 16,15,16,11,14,12
26110 DATA L2,9,L1,R,0,4,9
26120 DATA L2,11,L1,R,4,8,11
26130 DATA *1
26140 DATA L4,9
```

```
26150 DATA $
26160 DATA *2
26170 DATA °2
26180 DATA L2,9,L1,R,11,12,14
26190 DATA *
26200 DATA L3,16,L1,7,17,16
26210 DATA L3,14,L1,5,16,14
26220 DATA L3,12,L1,4,14,12
26230 DATA L2,11,L1,R,4,16,R
26240 DATA R,16,28,R2,15
26250 DATA 16,R2,15,16,15
26260 DATA 16,15,16,11,14,12
26270 DATA L2,9,L1,R,0,4,9
26280 DATA L2,11,L1,R,4,8,11
26290 DATA L2,12,L1,R,4,16,15
26300 DATA 16,15,16,11,14,12
26310 DATA L2,9,L1,R,0,4,9
26320 DATA L2,11,L1,R,4,12,11
26330 DATA °1
26340 DATA L2,9,L1,R,11,12,14
26350 DATA $
26360 DATA *2
26370 DATA L2,9,L1,R,0,4,9
26380 DATA 12,16,L4,21
26390 DATA fin
27000 REM *****************
27010 REM *Canal 1 Données*
27020 REM *****************
27030 DATA fin
28000 REM *****************
28010 REM *Canal 2 Données*
28020 REM *****************
28030 DATA ENV1,0-2
28040 DATA *
28050 DATA R2
28060 DATA R6
28070 DATA 9,16,21,R3
28080 DATA 4,16,20,R3
28090 DATA 9,16,21,R3
28100 DATA R6
28110 DATA 9,16,21,R3
28120 DATA 4,16,20,R3
28130 DATA °1
28140 DATA 9,16,21,R1
28150 DATA $
```

```
28160 DATA *2
28170 DATA °2
28180 DATA 9,16,21,R3
28190 DATA *
28200 DATA 12,19,24,R3
28210 DATA 7,19,23,R3
28220 DATA 9,16,21,R3
28230 DATA 4,16,28,R2,O0,4
28240 DATA 16,R,R,15,16,R
28250 DATA R,15,16,R3
28260 DATA R6,O-2
28270 DATA 9,16,21,R3
28280 DATA 4,16,20,R3
28290 DATA 9,16,21,R3
28300 DATA R6
28310 DATA 9,16,21,R3
28320 DATA 4,16,20,R3
28330 DATA °1
28340 DATA 9,16,21,R3
28350 DATA $
28360 DATA *2
28370 DATA °2
28380 DATA 9,16,21,R3
28390 DATA fin
```

Module 3.3.2 : Initialisation

Ce module est plus complexe que la normale, car il faut obtenir et traiter les données de la mélodie avant de passer à la partie du programme qui la joue réellement.



MODULE 3.3.2 : LIGNES 10000-10160

```
10000 REM ****************
10010 REM *Initialisation*
10020 REM ****************
10030 CLS:LOCATE 15,13:PRINT"Veuillez At
tendre"
10040 DIM a$(200,2),p$(2),p(2),d(2),s%(2
,500,4)
10050 FOR c=0 TO 2
10060 LET p=0
10070 READ a$(p,c)
```

```
10080 a$(p,c)=LOWER$(a$(p,c))
10090 IF a$(p,c)<>"fin" THEN p=p+1:GOTO
10070
10100 p$(c)="Y"
10110 NEXT c
10120 roc=0
10130 rno=0
10140 ul=20
10150 GOSUB 25000
10160 GOSUB 15000
```

Commentaires

Ligne 10040 : La complexité de la tâche est dûe à la variété de tableaux qui seront employés pendant le déroulement du programme. Nous les expliquerons au fur et à mesure de leur utilisation.

Lignes 10050-10110 : Les données sont lues dans les trois colonnes du tableau A$. Le tableau tel qu'il est défini ici peut contenir jusqu'à trois ensembles de 200 données, correspondant aux trois voix ou canaux du 464. Les données sont initialement chargées dans la première colonne. Lorsque le mot 'fin' est rencontré, la boucle continue à lire (READ) les données, mais commence à les placer dans la colonne suivante du tableau. Il est donc vital de terminer les données de chaque voix avec le mot 'fin' comme dans les specimens de modules DATA. Enfin, pour chaque voix, l'élément correspondant du tableau P$ reçoit la valeur Y pour indiquer, au moins initialement, qu'il faut jouer le canal particulier.

Ligne 10120 : La variable ROC permet de déterminer l'octave sur lequel la mélodie est basée. La mélodie DATA sera interprétée par rapport à cet octave de base.

Ligne 10130 : De même, RNO est la note de base. Si vous la changez pendant le déroulement du programme, la clé de la mélodie sera changée.

Ligne 10140 : La variable UL permet de stocker la longueur standard d'une note. Dans les données de la mélodie, chaque note sera un multiple de cette longueur standard.

Lignes 10150-10160 : Appellent deux sous-routines qui commencent à traiter les données de la mélodie sous une forme plus conforme au 464.

Test

Vous devez entrer deux lignes temporaires :

15000 RETURN
25000 RETURN

Puis exécutez le programme. Un temps assez long doit s'écouler avant le retour du curseur. Vous ne pouvez encore rien faire des données qui ont été collectées, mais vous saurez au moins que votre saisie ne contient pas d'erreur de syntaxe.

Module 3.3.3 : L'enveloppe de son

L'enveloppe d'une note est la forme qui guide sa montée à partir de rien, sa continuité et éventuellement son retour au silence. La plupart des sons et des instruments musicaux ont des enveloppes très particulières. La ou les commandes d'enveloppe du programme sont placées ici dans le programme pour vous rappeler qu'à bien des égards, elles font partie des données de la mélodie. Un changement d'enveloppe modifiera la mélodie presqu'autant qu'un changement de notes.

Nous n'allons pas approfondir ici le côté technique des enveloppes. Cela demanderait de nombreuses pages et vous apprendrez bien plus en faisant des essais avec différentes valeurs, lorsque le programme aura été entré.

MODULE 3.3.3 : LIGNES 25000-25040

```
25000 REM ***************************
25010 REM *Mettre ici les Commandes*
25015 REM *       ENV et ENT        *
25020 REM ***************************
25030 ENV 1,7,-1,10,8,-1,40
25040 RETURN
```

Module 3.3.4 : Traitement des données de la mélodie

Peut-être serez-vous surpris que nous ne jouions pas déjà la mélodie contenue dans les instructions DATA. Sachez que ce n'est pas pour tout de suite, car il faut d'abord traiter les données.

La raison en est que, en recherchant la façon la plus aisée d'enregistrer une mélodie, nous nous sommes éloignés du for-

mat nécessaire aux différentes commandes de son. La notation utilisée dans les instructions DATA, comme vous le verrez lorsque nous analyserons les différentes commandes qu'elles contiennent, est facile à mémoriser et à utiliser, mais il faut la traduire avant de l'introduire dans la commande SOUND du 464. Quelle que soit la rapidité du BASIC du 464, la tâche consistant à traduire les commandes jusqu'à trois voix et à les jouer simultanément, est au-delà de ses possibilités. Ce serait théoriquement possible, mais la vitesse à laquelle une mélodie pourrait être jouée serait limitée de façon inacceptable.

Nous avons donc choisi de faire d'abord toutes les traductions nécessaires, de stocker les paramètres de chacune des notes à jouer dans le tableau S%, et alors seulement de jouer la mélodie, à partir de S%.

MODULE 3.3.4 : LIGNES 15000-15140

```
15000 REM ************************
15010 REM *Compilation des données*
15020 REM ************************
15030 FOR c=0 TO 2
15040 l=1:v=12:o=0:ev=0:et=0:p=0:r=1
15050 n=0
15060 WHILE a$(p,c)<>"fin"
15070 FOR i=1 TO 8
15080 IF LEFT$(a$(p,c),1)<>MID$("ovl*'§e
r",i,1) THEN NEXT i
15090 ON i GOSUB 16000,17000,18000,19000
,20000,21000,22000,23000,24000
15100 p=p+1
15110 WEND
15120 d(c)=n
15130 NEXT c
15140 RETURN
```

Commentaires

Lignes 15030-15130 : L'opération est effectuée pour les trois voix, ou canaux, du 464..

Ligne 15040 : Les variables de cette ligne seront utilisées de la façon suivante :

L - longueur de note
V - volume
O - octave
EV - enveloppe de volume

ET - enveloppe de ton
P - position dans la séquence des commandes pour une voix
N - nombre de notes jouées

Lignes 15070-15090 : Similaire à la technique MENU$ utilisé dans le programme Caractères. Dans cette boucle, la commande contenue dans A% est comparée avec les commandes permises, qui se trouvent dans la chaîne de la ligne 15080. Si la première lettre de la commande correspond à l'une des lettres de la chaîne, la variable I de la boucle enregistre la position de la commande et cette donnée sera utilisée par ON...GOSUB. Nous verrons les effets des commandes individuelles dans les modules qui leur sont consacrés. Si la première lettre de la commande n'est *pas* l'une des lettres de la chaîne, la commande est considérée comme une valeur de note.

Ligne 15120 : Le nombre de notes pour chaque voix est stocké dans le tableau D.

Test

Le seul test, si l'on peut dire, consiste à placer des RETURN sur toutes les destinations spécifiées à la ligne 15090. Le programme sera ensuite exécuté pour que vous puissiez voir qu'il n'y a pas d'erreur de syntaxe. Le fait d'ajouter ces lignes RETURN vous permet aussi d'exécuter le programme chaque fois que vous appelez l'un des modules suivants, qui traitent les données de la mélodie.

Module 3.3.5 : La valeur d'une note

L'essentiel de la mélodie sera bien entendu constitué de notes et ce module a pour rôle de prendre les représentations des notes dans les données de la mélodie et de les rendre compréhensibles.

Les valeurs des notes elles-mêmes sont entrées d'après le système à 12 tons. En musique occidentale, un octave est divisé en huit tons entiers et quatre demi-tons, soit 12 notes en tout, de sorte que toute note d'un octave peut être représentée par un nombre de 0 à 11. Comme il y a une relation mathématique nette entre les notes d'un octave ou même les notes d'octaves différents, il est tout à fait possible d'aller au-delà de la gamme de 0 à 11, 12 devenant la première note de l'octave supérieur suivant, 24 la première note du suivant, et ainsi de suite. Il ne reste plus à ce module que le travail fastidieux de traduction.

MODULE 3.3.5 : LIGNES 24000-24070

```
24000 REM ******
24010 REM *Note*
24020 REM ******
24030 fr=440*(2^(o+(VAL(a$(p,c))-9)/12))
24040 tp=ROUND(125000/fr)
24050 s%(c,n,0)=tp:s%(c,n,1)=1:s%(c,n,2)
=v:s%(c,n,3)=ev:s%(c,n,4)=et
24060 n=n+1
24070 RETURN
```

Commentaires

Ligne 24030 : Cette formule traduit la valeur de note (en tenant compte de la valeur de l'octave, 0, que nous verrons plus loin), en une valeur de fréquence.

Ligne 24040 : La valeur de fréquence ayant été obtenue, cette ligne la traduit en une valeur capable d'agir sur la commande SOUND. Rien de spécial dans cette méthode ; tout simplement, le 464 est conçu pour fonctionner sur 125000/TRUE FREQUENCY.

Ligne 24050 : La présence d'une valeur de note dans les données de la mélodie est considérée comme une commande d'exécution d'une note. Donc les différentes données qui vont constituer une commande SOUND sont transférées à partir des variables où elles se trouvent, dans une ligne du tableau S%, qui sera éventuellement utilisée pour jouer la mélodie.

Module 3.3.6 : Changement de l'octave

Vous avez sûrement remarqué, que dans la formule d'extraction de la valeur d'une note, la valeur de l'octave (O) est prise en compte. Cela permet d'entrer les valeurs des notes avec plus de souplesse. La première note du deuxième octave peut être entrée avec la valeur 12, ou en prenant l'octave 1, spécifier la valeur de note 0. Pour spécifier un changement d'octave, il faut inclure un O suivi d'un nombre.

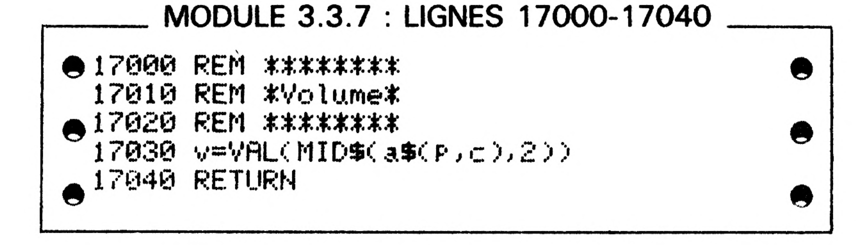
MODULE 3.3.6 : LIGNES 16000-16040

```
16000 REM ********
16010 REM *Octave*
16020 REM ********
```

```
16030 o=VAL(MID$(a$(p,c),2))
16040 RETURN
```

Module 3.3.7 : Spécification du volume

Pour spécifier le volume, incluez V suivi d'un réglage de volume correct, dans les données de la mélodie.



MODULE 3.3.7 : LIGNES 17000-17040

```
17000 REM ********
17010 REM *Volume*
17020 REM ********
17030 v=VAL(MID$(a$(p,c),2))
17040 RETURN
```

Module 3.3.8 : Définition de la longueur de note

Pour spécifier la longueur de note, incluez L suivi d'une valeur, dans les données de la mélodie. La longueur entrée dans les données de la mélodie sera multipliée par la longueur de note de base définie dans le module d'initialisation.

MODULE 3.3.8 : LIGNES 18000-18040

```
18000 REM **********
18010 REM *Longueur*
18020 REM **********
18030 l=VAL(MID$(a$(p,c),2))
18040 RETURN
```

Module 3.3.9 : Répétition d'une partie de la mélodie

La plupart des mélodies contiennent des répétitions; en fait, une mélodie sans répétition ne serait pas facile à écouter. Selon les cas, cette répétition concerne de petites ou de grandes parties de la mélodie. Pour gagner de la place, il est judicieux

de prévoir le repérage d'une partie de la mélodie qui doit être jouée deux fois consécutivement. Dans ce programme, cette section commence et se termine par un astérisque suivi du nombre de répétitions de cette partie.

MODULE 3.3.9 : LIGNES 19000-19090

```
19000 REM ***********
19010 REM *Répétition*
19020 REM ***********
19030 IF a$(p,c)="*" THEN r=1:RETURN
19040 IF VAL(MID$(a$(p,c),2))=r THEN RET
URN
19050 r=r+1
19060 WHILE a$(p,c)<>"*"
19070 p=p-1:IF p=-1 THEN RETURN
19080 WEND
19090 RETURN
```

Commentaires

Ligne 19030 : Pour atteindre ce module, la commande des données de la mélodie doit commencer par un astérisque. Si l'astérisque est seul, la variable R, qui stocke le nombre de répétitions de la partie, reçoit la valeur 1. Le programme principal reprend alors la main pour la traduction de la mélodie.

Ligne 19040 : Si l'astérisque est suivi d'un nombre, c'est la fin d'une partie. La valeur de R qui indique le nombre de répétitions de la partie, est comparée au nombre suivant l'astérisque, qui indique le nombre de répétitions qui auraient *dû* avoir lieu. Si les deux nombres sont identiques, le programme se poursuit au-delà de cette partie.

Lignes 19050-19080 : Le nombre de répétitions enregistré est augmenté et le programme revient au début de la partie pour la répéter, avant de revenir au programme principal.

Module 3.3.10 : Variation des parties répétées

Souvent, lorsqu'une partie de mélodie est répétée, il y a de légères modifications. La deuxième répétition peut se terminer sur une suite de notes hautes, tandis que la première répétition se terminait par des notes basses. Ce module vous permet de spécifier que, dans une partie répétée, un certain nombre de notes

ne doivent être incluses que pendant l'une des répétitions. Ainsi, vous pouvez répéter une partie deux fois avec deux conclusions différentes. Dans ce programme, nous marquons le début de la sous-section avec un symbole [suivi d'un nombre représentant le numéro de la répétition pendant laquelle la sous-section sera jouée, et la fin de la sous-section par |.

MODULE 3.3.10 : LIGNES 20000-21030

```
20000 REM ****************************
20010 REM *A la nième répétition de Do*
20020 REM ****************************
20030 IF VAL(MID$(a$(p,c),2))=r THEN RET
URN
20040 WHILE a$(p,c)<>"$" AND a$(p+1,c)<>
"fin"
20050 p=p+1
20060 WEND
20070 RETURN
21000 REM ***********
21010 REM *Fin de Do*
21020 REM ***********
21030 RETURN
```

Commentaires

Ligne 20030 : Ce point n'est atteint que si le programme a rencontré un crochet gauche dans le programme. La ligne examine le nombre suivant le [. Si sa valeur est la même que celle de R, qui indique le nombre de fois que la partie en cours a été répétée, le programme peut continuer à jouer les notes suivant le].

Lignes 20040-20060 : Cette boucle est utilisée si la sous-section ne doit pas être jouée pendant cette répétition ; elle recherche simplement après la sous-section jusqu'à ce qu'elle trouve le symbole] de fin.

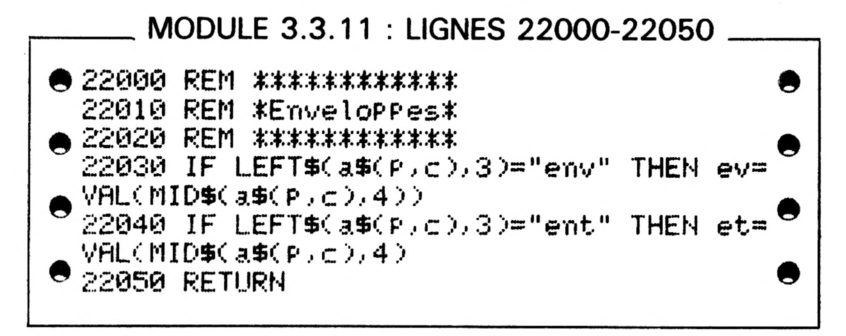
Ligne 21030 : Lorsque un] est rencontré, cette ligne permet simplement au programme de continuer.

Module 3.3.11 : Changement d'enveloppes

Vous avez déjà créé une enveloppe, mais le 464 peut en définir jusqu'à 16 pour le volume et le ton. En plaçant ENV ou ENT

suivi (sans espace) du nombre correct d'une enveloppe que vous avez définie dans le module à 25000, vous changez l'enveloppe utilisée par la commande SOUND.

MODULE 3.3.11 : LIGNES 22000-22050

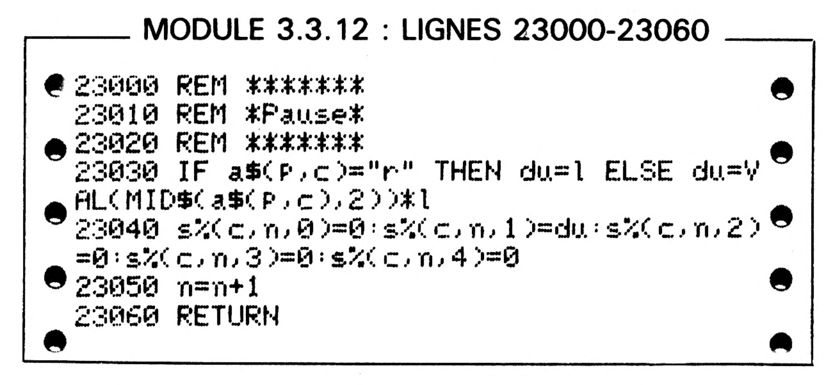


```
22000 REM ***********
22010 REM *Enveloppes*
22020 REM ***********
22030 IF LEFT$(a$(p,c),3)="env" THEN ev=
VAL(MID$(a$(p,c),4))
22040 IF LEFT$(a$(p,c),3)="ent" THEN et=
VAL(MID$(a$(p,c),4)
22050 RETURN
```

Module 3.3.12 : Création d'un silence

Dans une mélodie, les silences sont aussi importants que les sons, donc nous devons pouvoir en créer. Dans ce programme, nous l'obtenons en incluant R dans les données de la mélodie. Un seul R crée une pause de la longueur d'une unité de note, tandis que R suivi d'un nombre crée une pause d'une longueur égale au nombre.

MODULE 3.3.12 : LIGNES 23000-23060



```
23000 REM *******
23010 REM *Pause*
23020 REM *******
23030 IF a$(p,c)="r" THEN du=1 ELSE du=V
AL(MID$(a$(p,c),2))*1
23040 s%(c,n,0)=0:s%(c,n,1)=du:s%(c,n,2)
=0:s%(c,n,3)=0:s%(c,n,4)=0
23050 n=n+1
23060 RETURN
```

Test

Si vous ne l'avez pas fait pendant l'entrée des modules individuels, vous pouvez à présent exécuter le programme. Chacune des sections sera appelée pendant l'analyse des données de la mélodie.

Module 3.3.13 : Le menu de programme

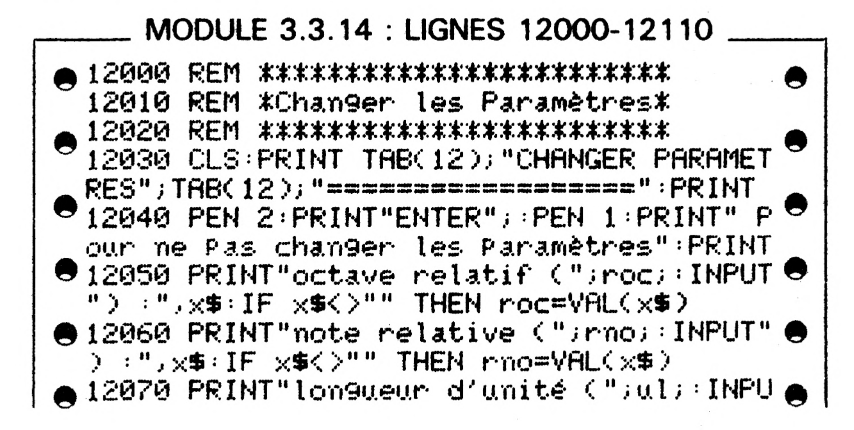
Il s'agit d'un menu qui permet simplement d'accéder aux autres fonctions du programme.

MODULE 3.3.13 : LIGNES 11000-11140

```
11000 REM **********
11010 REM *Controle*
11020 REM **********
11030 WHILE o<>3
11040 CLS:PRINT TAB(18);"MUSIQUE";TAB(18
);"======="
11050 PRINT:PRINT"Options:":PRINT
11060 PRINT"1) Changer les Paramètres"
11070 PRINT"2) Jouer la musique"
11080 PRINT"3) Fin"
11090 PRINT:INPUT"Entrer l'option: ",o
11100 ON o GOSUB 12000,13000
11110 WEND
11120 CLS
11130 LIST 25000-
11140 END
```

Module 3.3.14 : Changement des paramètres de la mélodie

Lorsqu'on en arrive à jouer la mélodie, plusieurs paramètres sont utilisés, que vous pouvez modifier avec ce module.


MODULE 3.3.14 : LIGNES 12000-12110

```
12000 REM ************************
12010 REM *Changer les Paramètres*
12020 REM ************************
12030 CLS:PRINT TAB(12);"CHANGER PARAMET
RES";TAB(12);"==================":PRINT
12040 PEN 2:PRINT"ENTER";:PEN 1:PRINT" P
our ne Pas changer les Paramètres":PRINT
12050 PRINT"octave relatif (";roc;:INPUT
") :",x$:IF x$<>"" THEN roc=VAL(x$)
12060 PRINT"note relative (";rno;:INPUT"
) :",x$:IF x$<>"" THEN rno=VAL(x$)
12070 PRINT"longueur d'unité (";ul;:INPU
```

```
T"):",x$:IF x$<>"" THEN ul=VAL(x$)
12080 FOR c=0 TO 2
12090 PRINT"canal";c;"(";p$(c);:INPUT")
:",x$:IF x$<>"" THEN p$(c)=UPPER$(x$)
12100 NEXT c
12110 RETURN
```

Commentaires

Ligne 12050 : L'octave relatif est la base à partir de laquelle la note à jouer est calculée.

Ligne 12060 : Avec l'octave de base, les notes jouées peuvent être calculées sur la base d'une note particulière, ce qui change la clé.

Ligne 12070 : Unité utilisée pour jouer une note. Une valeur plus élevée ralentit la musique.

Lignes 12080-12100 : Toutes les voix ne doivent pas être jouées; cette boucle permet d'en exclure certaines.

Test

Si vous exécutez le programme, après la pause permettant de recalculer les données de la mélodie, le menu doit apparaître. Vous pouvez sélectionner l'option 1 et spécifier les paramètres de votre choix.

Module 3.3.15 : Contrôle de la musique

Nous passons aux deux modules qui vont réaliser l'essentiel du programme : jouer une mélodie. Le premier est un module de contrôle qui distribue le travail et permet de terminer la mélodie, le module suivant joue les notes.

MODULE 3.3.15 : LIGNES 13000-13160

```
13000 REM *****************
13010 REM *Jouer la musique*
13020 REM *****************
13030 rtp=2^-(roc+rno/12)
13040 FOR c=0 TO 2
13050 p(c)=0
13060 NEXT c
13070 SOUND 199,0,0
```

```
13080 GOSUB 14000
13090 GOSUB 14000
13100 GOSUB 14000
13110 RELEASE 7
13120 fait=0
13130 WHILE NOT fait
13140 GOSUB 14000
13150 WEND
13160 RETURN
```

Commentaires

Ligne 13030 : Base à partir de laquelle la mélodie est jouée.

Lignes 13040-13060 : Le tableau de comptage P, pour les trois voix, est mis à 0.

Ligne 13070 : Cette instruction ne joue pas une note ; c'est une commande de service qui joue trois rôles :

1) Supprimer toutes les notes qui pourraient se trouver dans la file d'attente depuis l'utilisation précédente.
2) Spécifier que les commandes doivent être envoyées aux trois voix.
3) Empêcher que des notes ne soient jouées.

Le nombre 199 n'a pas vraiment de signification : c'est simplement le moyen de donner la valeur 1 à certains chiffres binaires ou bits. Dans le cas de 199, les bits 0, 1, 2, 6 et 7 sont actifs. Vous trouverez au chapitre 6 de ce manuel des détails sur les effets de ces bits.

Lignes 13080-13110 : Ces trois branchements au module suivant créent une file d'attente de notes attendant d'être jouées. Ainsi, les légers retards pouvant provenir de la boucle qui joue la partie principale de la mélodie n'influenceront pas la régularité de la mélodie elle-même. La commande RELEASE à la ligne 13110 fait commencer les trois voix simultanément dès que les premières notes de la file d'attente ont été formées. Cette possibilité de file d'attente du 464 est l'une de ses caractéristiques musicales les plus remarquables. Sur la plupart des autres micros, le rythme d'exécution des notes est extrêmement difficile à maîtriser, car tout retard dans le traitement d'une note a un effet négatif sur le rythme. Le 464 permet d'éliminer ce problème ; le programme en devient à la fois plus simple et efficace.

Lignes 13120-13140 : Le reste de l'exécution de la mélodie consiste tout simplement à appeler le module suivant continuellement, jusqu'à ce que la variable FAIT, définie par le module suivant, indique que les données de la mélodie sont épuisées.

Module 3.3.16 : Exécution des notes

C'est le dernier module qui joue les notes pour les trois voix.

MODULE 3.3.16 : LIGNES 14000-14120

```
14000 REM ******************
14010 REM *Examen des canaux*
14020 REM ******************
14030 fait=-1
14040 FOR c=0 TO 2
14050 IF p(c)=d(c) OR p$(c)="N" THEN GOT
O 14110
14060 fait=0
14070 IF (SQ(2^c) AND 7)=0 THEN GOTO 141
10
14080 n=p(c)
14090 SOUND 2^c,s%(c,n,0)*rtp,s%(c,n,1)*
ul,s%(c,n,2),s%(c,n,3),s%(c,n,4)
14100 p(c)=p(c)+1
14110 NEXT c
14120 RETURN
```

Commentaires

Lignes 14040-14050 et 14110 : Que les trois voix doivent être jouées ou non, cette boucle va les examiner toutes les trois. Toutefois, la ligne 14050 détecte si une voix particulière a épuisé toutes ses données ou si l'utilisateur a redéfini les paramètres afin d'exclure une voix particulière. S'il y a au moins un accès à la partie principale de la boucle (c'est-à-dire s'il reste des notes à jouer pour au moins une voix), la valeur de FAIT est mise à 0.

Ligne 14070 : La fonction SQ permet de déterminer s'il est possible de placer une autre note dans la file d'attente de la voix en question.

Lignes 14080-14100 : S'il *est* possible d'ajouter une autre note à la file d'attente, le prochain ensemble de données de S% sert de base à la commande SOUND. A noter que cela n'exécute pas immédiatement une note, mais ne fait que l'ajouter à la file d'attente des notes en attente d'exécution. Pour finir, le compteur de la voix en question est incrémenté.

Test

Enfin, vous allez pouvoir faire un test réaliste du programme. Pour cela, il vous suffit de le lancer et, lorsque le menu apparaît, de spécifier l'option 2. Vous saurez immédiatement s'il fonctionne.

4

Ça se complique

A ce stade de votre progression, vous devez vous sentir plus familiarisé avec les possibilités du 464 et de certaines de ses techniques. Il est temps de passer à certains programmes plus importants qui vont permettre à votre 464 de se livrer aux tâches préférées des micro-ordinateurs : le traitement, le tri et la recherche d'informations.

Voici les programmes de ce chapitre :

UNIFILE : Un puissant système de classement personnel capable de stocker et de relire immédiatement une grande variété d'informations.

NNOMBRE : Un programme qui crée un dictionnaire de Noms et de Nombres pour toutes sortes d'applications. Vous pourrez créer des factures, gérer un stock, ou même compter les calories des menus quotidiens.

TEXTE : Un programme de traitement de texte simple exécuté en BASIC.

MULTIQ : Un générateur de test à choix multiples.

PROGRAMME 4.1 : UNIFILE

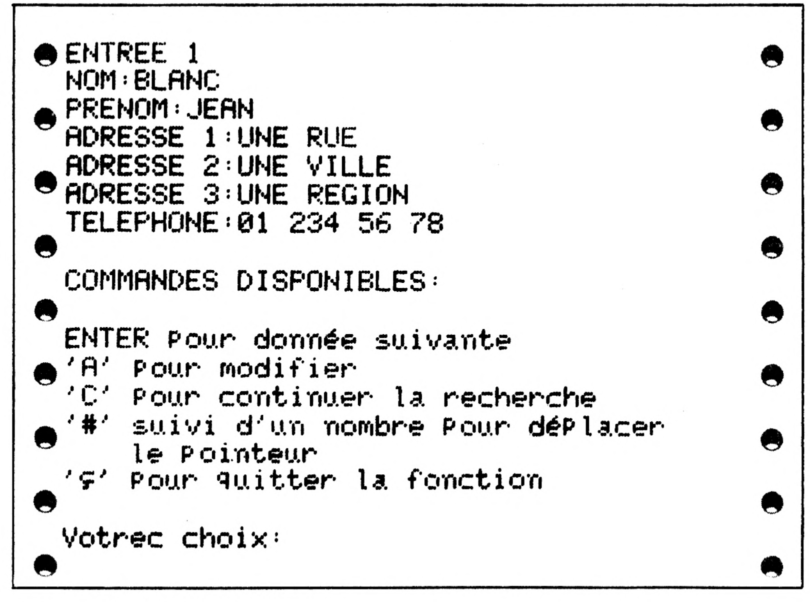
Fonction du programme

Unifile est une merveille de programme, fruit de plusieurs années de développement dans les livres «Activités sur». Les lecteurs des ouvrages précédents nous ont déclaré les utiliser dans leurs activités professionnelles, pour apprendre aux élèves la technique de traitement de l'information par des micros, pour aider des clubs et des associations bénévoles, ou simplement pour leur gestion familiale.

Ce programme présente les nouveaux concepts suivants :

1) Recherche binaire
2) Regroupement d'éléments en chaînes continues
3) Lancement d'un programme sans effacer les tableaux.

Module 4.1.1 : Création de la structure du fichier



Unifile en Mode Recherche

De nombreux ouvrages destinés aux utilisateurs d'ordinateurs domestiques proposent des programmes de gestion de fichier de qualité inférieure, manquant de souplesse d'utilisation. La

nature du programme impose à l'utilisateur, chaque fois qu'il stocke une donnée, de le faire sous les intitulés : Nom, Adresse, Numéro de téléphone ou autres structures similaires. L'intérêt d'Unifile est que, tout en autorisant une telle structure, il permet aussi de créer d'autres fichiers sous des structures différentes; peut-être avec un intitulé, peut-être avec dix, sans modifier en quoi que ce soit le programme lui-même. Unifile est un programme du type »caméléon» : prêt à s'adapter à une grande variété d'utilisations, réagissant avec l'utilisateur selon la tâche à effectuer.

Ce premier module a pour but d'initialiser quelques variables, mais aussi et surtout de vous permettre de bâtir le fichier d'origine tel que vous le souhaitez.

MODULE 4.1.1 : LIGNES 20000-20110

```
20000 REM ***********************
20010 REM *Création d'un fichier*
20020 REM ***********************
20030 CLS:PRINT TAB(12);"Nouveau Fichier
":PRINT TAB(12);"==============="
20040 PRINT:INPUT"Etes-vous certain (o/n
)";r$:IF LEFT$(UPPER$(r$),1)="N" THEN RE
TURN
20050 CLEAR:DIM tableau$(1000)
20060 PRINT:INPUT"Combien de données dan
s chaque entrée";x
20070 DIM item$(x-1),ptr(x-1)
20080 PRINT:FOR i=0 TO x-1
20090 PRINT"Nom de la donnée #";i+1;:INP
UT":",item$(i)
20095 item$(i)=UPPER$(item$(i))
20100 NEXT i
20110 in=-1:GOTO 11000
```

Commentaires

Lignes 20030-20040 : La création de nouveaux fichiers va effacer tout le contenu de la mémoire. Donc, vous avez la possibilité de ne pas poursuivre.

Ligne 20050 : L'information destinée à Unifile sera stockee dans le tableau TABLEAU$. Dans cette ligne, il est dimensionné pour autoriser 1001 entrées dans le fichier. Vous pouvez porter ce nombre à 2000 ou plus. Il faut simplement savoir que chaque élément du tableau, avant d'être utilisé, occupe

trois octets de mémoire. Donc, si vous dimensionnez le tableau avec 5000 données, vous utiliserez 15000 octets de mémoire, soit un tiers de la mémoire disponible, avant d'avoir stocké une quelconque information. A vous d'évaluer : si vous pensez avoir moins de 1000 entrées dans un fichier, il est peut-être bon de laisser l'instruction DIM en l'état et d'épargner un peu de mémoire. A noter aussi que, comme nous utilisons un élément d'un tableau pour stocker chaque entrée, la longueur maximale d'une simple entrée y compris ses caractères de séparation, est de 255 caractères : la longueur maximale d'une chaîne pouvant être traitée par le 464.

Lignes 20060-20100 : Dans ces lignes réside partiellement le secret de la souplesse d'Unifile. Pour chaque *entrée,* que vous pouvez imaginer comme une fiche de classement, vous pouvez définir le nombre de données qui apparaîtront dans l'entrée. S'il s'agissait de votre collection de disques, vous pourriez utiliser les intitulés : PISTE, ALBUM, COMPOSITEUR, INTERPRETE, DUREE. Dans ce cas vous spécifieriez cinq données puis vous entreriez leur nom. Dans le futur, chaque fois que vous utiliserez Unifile pour stocker ou lire des informations sur votre collection de disques, vous devrez donner une donnée correspondant à chacun de ces intitulés. Pour employer un langage plus technique, ce que nous appelons une *entrée* pour désigner la fiche de classement et des *données* pour chacun des intitulés, s'appelerait en jargon informatique des *enregistrements* et des *zones* (ou champs) respectivement. Pendant les divers commentaires sur le programme, nous expliquerons l'utilisation des variables définies ici.

Module 4.1.2 : Mise en route du programme

La possibilité de créer un nouveau fichier n'est pas utilisée chaque fois que le programme est exécuté. Le plus souvent, il s'agira de recharger des données à partir d'une cassette. Ce module de mise en route vous permet simplement de choisir l'une de ces deux options.

MODULE 4.1.2 : LIGNES 10000-10150

```
10000 REM ***************
10010 REM *Mise en route*
10020 REM ***************
10030 WHILE NOT in
10040 CLS
```

```
10050 PRINT TAB(12);"CREATION D'UN FICHI
ER"
10060 PRINT TAB(12);"==================
=="
10070 PRINT
10080 PRINT"COMMANDES DISPONIBLES:"
10090 PRINT
10100 PRINT"1) Créer nouveau fichier"
10110 PRINT"2) Lecture sur cassette"
10120 PRINT
10130 INPUT"Votre choix: ",z
10140 ON z GOSUB 20000,21000
10150 WEND
```

Commentaires

Lignes 10030 et 10150 : Ces deux lignes représentent une technique très utile que nous vous conseillons dans vos propres programmes. Lorsque le programme est initialisé par le premier module entré, une variable IN (abréviation de INitialisation) a reçu la valeur -1. Cette boucle explique pourquoi. Si IN a la valeur -1 lorsque la ligne 10030 est atteinte, cette boucle n'est pas exécutée. L'opérateur NOT associé à WHILE à la ligne 10030 ne permet d'entrer dans la boucle que si la valeur de IN est 0. Donc, si vous arrêtez le programme et si vous le relancez avec GOTO 10000 (ou GOTO 1 si vous avez inclus le petit module SAVE recommandé dans le premier programme de ce livre), aucune des données en mémoire n'est détruite.

Test

Entrez une ligne temporaire :

11000 STOP

Exécutez le programme et spécifiez que vous voulez créer un nouveau fichier.

Suivez les indications et répondez en donnant quelques nombres et quelques noms vraisemblables. Puis le programme s'arrête. Entrez :

GOTO 10000[ENTER]

Le programme s'arrête immédiatement : il a détecté un fichier en mémoire, bien qu'il soit vide, et il a contourné le module à 10000.

Module 4.1.3 : Le menu

Il s'agit d'un menu standard, mais qui ne permet pas d'arrêter le programme au moyen de ON ERROR et de la touche ESC. Unifile est un programme complexe et le fait de l'arrêter en plein milieu pourrait altérer une partie de ses informations, du fait qu'une opération importante n'est pas allée à son terme. Pour mettre fin à Unifile, vous devez *toujours* revenir à ce menu et utiliser l'option 6.

MODULE 4.1.3 : LIGNES 11000-11210

```
11000 REM ****************
11010 REM *Menu Principal*
11020 REM ****************
11030 WHILE NOT fait
11040 CLS
11050 PRINT TAB(12);"Création d'un fichi
er"
11060 PRINT TAB(12);"==================
=="
11070 PRINT
11080 PRINT"COMMANDES DISPONIBLES:"
11090 PRINT
11100 PRINT"1) Créer nouveau fichier"
11110 PRINT"2) Charger fichier cassette"
11120 PRINT"3) Entrer des informations"
11130 PRINT"4) Rechercher/Afficher/Modif
ier"
11140 PRINT"5) Sauvegarde fichier sur ca
ssette"
11150 PRINT"6) Arreter"
11160 PRINT
11170 INPUT"Votre choix: ",z
11180 ON z GOSUB 20000,21000,12000,17000
,18000,22000
11190 WEND
11200 fait=0
11210 END
```

Module 4.1.4 : Arrêt du programme

Ce petit module affiche le message de fin pour le programme et définit la valeur de la variable FAIT pour informer le menu que le programme doit s'arrêter.

MODULE 4.1.4 : LIGNES 22000-22060

```
22000 REM *******
22010 REM *Arret*
22020 REM *******
22030 fait=-1
22040 CLS:LOCATE 11,13
22050 PRINT"SYSTEME DE FICHIERS FERME":P
RINT:PRINT
22060 RETURN
25000 OPENIN "Création d'un fichier"
25010 INPUT t$
25020 PRINT t$
25030 GOTO 25010
```

Test

Entrez :

GOTO 11000[ENTER]

Le menu s'affiche. Spécifiez l'option 6 qui met fin au programme.

Module 4.1.5 : Sauvegarde des données sur cassette

Lorsque vous écrirez des programmes plus complexes, qu'ils soient de cet ouvrage ou vos propres programmes, vous souhaiterez entrer le module DATA FILE (fichier de données) le plus tôt possible. En effet, ce n'est qu'avec de grandes quantités de données que vous pouvez tester de façon approfondie Unifile.

Comme vous risquez de faire des erreurs et que les données peuvent être altérées, vous serez contraint de réentrer plusieurs fois les mêmes données au fur et à mesure du développement du programme. Pour pallier cet inconvénient, il suffit de stocker ces données sur cassette dès que possible et de relire les informations à partir de la cassette, pendant les tests successifs.

Ce module vous permet de stocker des données et vous présente aussi une idée dont nous avons parlé mais que nous n'avons pas encore utilisée : permettre à un programme de lire ou de stocker des fichiers ayant une grande diversité de noms.

MODULE 4.1.5 : LIGNES 18000-18080

```
18000 REM *************************
18010 REM *Sauvegarde D'un fichier*
18020 REM *************************
18030 CLS:PRINT TAB(12); "Sauvegarde Fic
hier":PRINT TAB(12);"================"
18040 PRINT:INPUT"Nom de sauvegarde: ",f
i$
18050 PRINT:OPENOUT fi$:PRINT#9,it:PRINT
#9,x
18060 FOR i=0 TO it-1:PRINT#9,tableau$(i
):NEXT i
18070 FOR i=0 TO x-1:PRINT#9,item$(i):NE
XT i
18080 CLOSEOUT:RETURN
```

Commentaires

Lignes 18030-18040 : Le nom du fichier est spécifié de façon interactive.

Lignes 18050-18070 : Le nombre d'entrées contenues par Unifile à n'importe quel moment est stocké dans la variable IT, tandis que X enregistre le nombre de données de chaque entrée. Les lignes enregistrent tous les éléments utiles provenant de TABLEAU$ et les noms d'intitulés provenant de DONNEE$.

Module 4.1.6 : Chargement des données à partir de la cassette

Nous allons à présent relire les données déjà enregistrées sur cassette. C'est moins simple qu'il n'y paraît car, alors que, lors de la sauvegarde, tous les tableaux étaient déjà correctement dimensionnés pour le fichier en mémoire, lorsqu'un fichier est chargé à partir d'une cassette, il faut réinitialiser tous les tableaux. C'est pourquoi nous avons placé les variables IT et X au début du fichier, pour qu'elles puissent être d'abord relues en mémoire, puis utilisées pour dimensionner les tableaux tout comme au premier module, où elles ont été entrées par l'utilisateur et non lues automatiquement.

MODULE 4.1.6 : LIGNES 21000-21140

```
21000 REM ************************
21010 REM *Chargement d'un fichier*
21020 REM ************************
21030 CLS:PRINT TAB(12);"Chargement fich
ier":PRINT TAB(12);"=================="
21040 PRINT:INPUT"Etes-vous certain (o/n
)";r$:IF LEFT$(UPPER$(r$),1)="N" THEN RE
TURN
21050 CLEAR:DIM tableau$(1000)
21060 PRINT:INPUT"Nom du fichier à sauve
rgarder: ",fi$
21070 PRINT:OPENIN fi$:INPUT #9,it,x:DIM
 item$(x-1),ptr(x-1)
21080 FOR i=0 TO it-1
21090 INPUT #9,tableau$(i)
21100 NEXT i
21110 FOR i=0 TO x-1
21120 INPUT #9,item$(i)
21130 NEXT i
21140 in=-1:CLOSEIN:GOTO 11000
```

Commentaires

Ligne 21140 : Peut-être l'instruction GOTO à la fin d'une sous-routine vous surprend-t-elle. Elle ne peut pas être remplacée par un RETURN, car pendant le déroulement du module, la mémoire a été effacée pour permettre le redimensionnement des tableaux. Cet effacement a également affecté toute mémorisation associée à la commande GOSUB d'origine, et donc RETURN ne trouvant plus de destination, génèrerait simplement un message d'erreur.

Test

Exécutez le programme en spécifiant une création de nouveau fichier. Créez quatre données, nommées UN, DEUX, etc. Lorsque le menu apparaît, choisissez l'option 5 puis spécifiez un nom de fichier tel que UNIDATA. Avant de commencer l'enregistrement, assurez-vous que le lecteur contient une cassette correcte. Lorsque le programme revient au menu, arrêtez-le avec l'option 6, puis lancez-le à nouveau. Cette fois-ci, spécifiez un chargement à partir de la cassette, donnez le nom du fichier de données et lisez (replay) ce fichier lorsque le programme vous le demande (après avoir rebobiné la cassette bien

entendu). Lorsque le menu réapparaît, arrêtez le programme à nouveau. Entrez :

FOR I=0 TO X-1 :PRINT DONNEE$(I) :NEXT[ENTER]

vous devez obtenir l'affichage des quatre intitulés.

Module 4.1.7 : Une meilleure méthode de recherche

Dans ce module et les deux suivants, nous examinons comment ajouter une nouvelle entrée au fichier principal contenu dans TABLEAU$. L'essentiel de la méthode se trouve toutefois dans ce module qui permet à Unifile de trouver rapidement dans un grand fichier l'endroit correct pour insérer une nouvelle entrée, ou de rechercher rapidement la présence d'une entrée importante dans le fichier.

La méthode est appelée 'recherche binaire'. Elle réduit le temps de recherche dans tout programme contenant de longues listes de données ordonnées. Nous allons le voir avec l'exemple suivant.

Un fichier de 2000 noms a été créé. Nous voulons insérer un nouveau nom en respectant l'ordre alphabétique. Si nous trichons et examinons la liste des noms, nous savons que le nouveau nom 'YOUNGERS' doit aller dans la position 1731 du fichier, bien que l'ordinateur l'ignore encore.

Bien sûr, nous pouvons demander à l'ordinateur d'examiner les noms un par un à partir du début. Il commence avec 'ADAMS' et constate que 'YOUNGER' doit venir après, puis il passe à 'ADAMSON' et ainsi de suite. Après avoir examiné 1732 noms, il trouvera un nom tel que 'YOUNGMAN' qui doit se trouver après 'YOUNGER'. Ainsi, la position correcte a été trouvée.

Certes, cette méthode est sûre, mais nous aimerions bien réduire le nombre de comparaisons. Et bien, dans le cas de notre fichier de 2000 noms, 10 comparaisons vont suffire. Voici comment.

L'ordinateur commence la recherche en examinant le nom à la position 1024 du fichier, parce que 1024 est la plus grande puissance de deux (2^{10}) qui contienne dans le nombre total d'entrées (2000). Le nom de la position 1024 précède alphabétiquement 'YOUNGER', donc l'ordinateur ajoute 1024/2 ou 512 ou 2^9, à la position de recherche initiale, et arrive à 1536. A nouveau, le nom de cette position précède alphabétiquement

'YOUNGER' donc 256, ou 28, est ajouté à 1536, pour donner 1792. Ici, nous trouvons du nouveau. Le nom de la position 1792 suit dans l'ordre alphabétique 'YOUNGER', donc au lieu d'ajouter 128 ou 27, ce montant est soustrait de la position de recherche, pour donner 1664.

Et ainsi, la recherche se poursuit, en ajoutant ou en soustrayant des puissances décroissantes de 2 pour bâtir le profil de recherche suivant :

NUMERO DE COMPARAISON	POSITION	ACTION
1	1024	+512
2	1536	+256
3	1792	-128
4	1644	+64
5	1728	+32
6	1760	-16
7	1744	-8
8	1736	-4
9	1732	-2
10	1730	+1

Essayez vous-même avec différents nombres ou entrées, et différentes positions dans l'ordre. Vous constaterez que le résultat est toujours bon.

MODULE 4.1.7 : LIGNES 13000-13110

```
13000 REM ******************
13010 REM *Rcherche binaire*
13020 REM ******************
13030 IF it=0 THEN ss=0:RETURN
13040 Po=INT(LOG(it)/LOG(2)):ss=2^Po-1
13050 FOR i=Po-1 TO 0 STEP -1
13060 ss=ss+2^i*((tableau$(ss)>te$)-(tab
leau$(ss)<te$))
13070 IF ss<0 THEN ss=0
13080 IF ss>it-1 THEN ss=it-1
13090 NEXT i
13100 IF tableau$(ss)<te$ THEN ss=ss+1
13110 RETURN
```

Commentaires

Ligne 13030 : S'il n'y a pas d'entrées dans le fichier, le calcul entraîne une erreur. Cette ligne évite cette situation.

Ligne 13040 : Ces deux expressions trouvent la plus grande puissance de 2 qui pourra être contenue dans le nombre de rubriques du fichier, puis elles donnent cette valeur au pointeur de recherche (SS). Le '-1' de la deuxième expression prend en compte le fait que le tableau est numéroté à partir de 0, et non de 1.

Lignes 13050-13090 : Cette boucle conduit la recherche, avec des puissances décroissantes de 2. Le fichier principal est contenu dans TABLEAU$ et la nouvelle entrée dans TE$. La valeur du pointeur est définie par deux conditions logiques qui indiquent si TE$ est supérieur ou inférieur à l'entrée de TABLEAU$ (SS).

Ligne 13100 : Dans certains cas, la position obtenue sera inférieure de 1 à la position correcte. Dans ce cas, SS est augmenté de 1.

Module 4.1.8 : Insertion d'une donnée

Ce module insère la nouvelle entrée à la position indiquée par la variable SS. Pour cela, il suffit de tout déplacer d'une position vers le haut à partir de la position SS.

MODULE 4.1.8 : LIGNES 14000-14070

```
14000 REM ************************
14010 REM *Insertion d'une entrée*
14020 REM ************************
14030 IF it=0 THEN GOTO 14070
14040 FOR i=it TO ss+1 STEP -1
14050 tableau$(i)=tableau$(i-1)
14060 NEXT i
14070 tableau$(ss)=te$:it=it+1:RETURN
```

Module 4.1.9 : Saisie de nouvelles entrées dans le fichier

Après avoir entré les modules qui font le travail réel, nous allons voir celui qui permet de saisir des données dans le système de fichiers. Ce module vous invite à entrer le nombre correct de données, dans le bon ordre, pour chaque entrée, afin de combiner ces données dans une chaîne qui sera contenue

dans une ligne de TABLEAU$; puis il appelle les deux modules précédents pour placer les données ainsi saisies dans le fichier principal.

MODULE 4.1.9 : LIGNES 12000-12320

```
12000 REM ******************
12010 REM *Nouvelles entrées*
12020 REM ******************
12030 WHILE it<1000
12040 te$=""
12050 CLS
12060 PRINT TAB(10);"NOUVELLES ENTREES"
12070 PRINT TAB(10);"================="
12080 PRINT
12090 PRINT"Numéro de l'entrée";it+1
12100 PRINT
12110 PRINT"COMMANDES DISPONIBLES:"
12120 PRINT
12130 PRINT"Entrer la donnée spécifiée"
12140 PRINT"'ç' pour revenir au menu pri
ncipal"
12150 PRINT
12220 FOR i=0 TO x-1
12230 PRINT item$(i);:INPUT":",q$
12240 IF q$="ç" THEN RETURN
12250 te$=te$+UPPER$(q$)+"ç"
12260 NEXT i
12270 PRINT:PRINT"Attendez un instant ..
."
12280 GOSUB 13000:GOSUB 14000
12290 WEND
12300 CLS
12310 PRINT"*** DESOLE, PLUS D'ENTREES P
OSSIBLES ***"
12320 FOR i=1 TO 2000:NEXT i:RETURN
```

Commentaires

Lignes 12030 et 12290-12320 : Les entrées ne sont acceptées que s'il y a de la place pour les accueillir ; sinon, un message d'erreur apparaît. Si vous vouliez changer le nombre maximum d'entrées, vous devriez remplacer cette référence par 1000. Vous pourriez aussi la remplacer par une variable, à condition de définir cette dernière lors de l'initialisation du programme et de la sauvegarder lorsqu'un fichier particulier est stocké.

Lignes 12220-12260 : Ces lignes demandent l'entrée des données individuelles de la nouvelle entrée. Le nom de chaque donnée provient du tableau DONNEE$, créé par le module d'initialisation. Chaque donnée est entrée sous le nom Q$ et l'entrée des nouvelles données est terminée dès que Q$ est égal à ' ', quel que soit l'endroit. Si la donnée est différente de ' ', elle s'ajoute à TE$ qui contient l'entrée globale et un caractère ' ' est ajouté à la fin de la donnée. Ce caractère ' ' n'a rien à voir avec le précédent qui terminait la saisie de l'information, il indique simplement la limite entre deux données. Ainsi, une entrée du type

DURAND
JEAN
11 RUE BLANCHE
POISSY

serait stockée dans TABLEAU$ sous la forme :

DURAND JEAN 11 RUE BLANCHE POISSY

créant une 'chaîne compactée'. La raison du choix du signe ' ' est qu'il est peu probable qu'il figure réellement dans une entrée; si c'était le cas, choisissez un autre caractère de séparation. A noter que toutes les données entrées sont converties en majuscules. Nous verrons plus tard qu'il en est de même pour la recherche de données dans le fichier. Cela facilite l'entrée et la recherche dans le fichier, puisque le système ne distingue pas les majuscules des minuscules. Si vous *voulez* distinguer des majuscules des minuscules, cette possibilité peut être supprimée sans que le fonctionnement du programme en souffre. Toutefois, sachez que, pour le 464, toute lettre minuscule est considérée, dans l'ordre alphabétique, inférieure à toute lettre majuscule; donc le classement de votre fichier risque de paraître étrange.

Test

Vous pouvez à présent tester les trois derniers modules entrés. Lancez le programme et créez un fichier avec deux données par entrée, appelées 'UN' et 'DEUX'. Après avoir initialisé le programme et être revenu au menu principal, spécifiez l'option 3. L'affichage obtenu est celui du module en cours; il vous demande d'entrer la donnée 'UN'. Entrez 'AA1'. Le message se répète pour 'DEUX' et vous devez entrer 'AA2'. Répondez aux messages successifs par 'DD1', 'DD2', 'CC1', 'CC2', 'BB1', 'BB2'. Puis entrez ' ' qui vous ramène au menu principal. Choisissez l'option 6 pour arrêter le programme. Puis entrez :

FOR I=0 TO 3 : ? TABLEAU$(I) :NEXT[ENTER]

vous devez obtenir :

AA1 AA2
BB1 BB2
CC1 CC2
DD1 DD2

Si tout s'est bien passé, vous pouvez relancer le programme avec GOTO 10000 et utiliser l'option 3 pour stocker sur cassette les données entrées. Cela facilitera les tests suivants.

Module 4.1.10 : Identification des données dans une simple entrée

Avant de passer au module permettant la lecture et le traitement des données à partir du fichier, nous devons entrer ce module qui a pour tâche d'explorer une entrée et d'enregistrer la position de chaque caractère ' ' ou, en fait, la fin de chaque donnée dans l'entrée. Les valeurs sont stockées dans le tableau PTR et ne concernent que l'entrée en cours. Pour une autre entrée vous devrez appeler de nouveau ce module afin qu'il se livre à l'analyse.

MODULE 4.1.10 : LIGNES 19000-19080

```
19000 REM ******************************
19010 REM *Analyse de l'enregistrement*
19020 REM ******************************
19030 pp=0
19040 FOR i=0 TO x-1
19050 ptr(i)=INSTR(pp+1,tableau$(s1),"ç"
)
19060 pp=ptr(i)
19070 NEXT i
19080 RETURN
```

Commentaires

Ligne 19050 : Si l'on excepte le module de recherche binaire, où la variable SS est utilisée pour indiquer l'entrée examinée, le reste du programme utilise la variable S1 pour enregistrer la position dans TABLEAU$ de l'entrée en cours. Cette ligne a pour effet de rechercher chaque présence du caractère ' ', en commençant à un caractère après l'endroit où le dernier a été trouvé.

Module 4.1.11 : Recherche des données dans le fichier

Et nous voici au module qui donne au programme toute sa signification ; nous allons pouvoir lire les données stockées. Ce module permet de lire (ou d'extraire) les données par l'une des quatre méthodes suivantes :

1) Une à une, dans l'ordre à partir de la position en cours.
2) En sautant vers l'avant ou vers l'arrière d'un nombre de données spécifié.
3) En entrant une donnée-clé - la première donnée de l'entrée - pour une recherche rapide.
4) En recherchant toute présence d'une combinaison de caractères, où qu'elle se trouve dans une entrée.

MODULE 4.1.11 : LIGNES 17000-17430

```
17000 REM ***********
17010 REM *Recherche*
17020 REM ***********
17030 s1=0:CLS:PRINT TAB(14);"RECHERCHE"
:PRINT TAB(14);"=========":PRINT
17040 IF it=0 THEN PRINT"*** PAS ENCORE
D'ENTREE ***" ELSE GOTO 17070
17050 FOR i=1 TO 2000:NEXT i
17060 RETURN
17070 PRINT"COMMANDES DISPONIBLES:"
17080 PRINT:PRINT"Entrer donnée pour rec
herche normale"
17090 PRINT"Faire précéder de '*' pour r
echerche initiale"
17100 PEN 2:PRINT"ENTER";
17110 PEN 1:PRINT", pour première donnée
du fichier"
17120 te$="":PRINT:INPUT"Entrer commande
 de recherche: ",te$:te$=UPPER$(te$)
17130 IF LEFT$(te$,1)="*" THEN te$=MID$(
te$,2):GOSUB 13000:te$="":s1=ss
17140 p$="C":WHILE p$="C"
17150 IF te$="" THEN GOTO 17210
17160 ff=0:FOR i=s1 TO it-1
17170 pp=INSTR(tableau$(i),te$)
17180 IF pp<>0 THEN ff=1:s1=i:i=it-1
17190 NEXT i
17200 IF ff=0 THEN RETURN
17210 p$=""
```

```
17220 WHILE p$="" AND it>0
17230 IF s1>it-1 THEN s1=it-1
17240 IF s1<0 THEN s1=0
17250 GOSUB 19000
17260 CLS:PRINT"ENTREE ";s1+1;":":PRINT:
pp=0
17270 FOR i=0 TO x-1
17280 PRINT item$(i);":";MID$(tableau$(s
1),pp+1,ptr(i)-pp-1)
17290 pp=ptr(i):NEXT i
17300 PRINT:PRINT" COMMANDES DISPONIBLES
:"
17310 PRINT:PEN 2:PRINT"ENTER";:PEN 1:PR
INT" pour donnée suivante"
17320 PRINT"'A' pour modifier"
17330 PRINT"'C' pour continuer la recher
che"
17340 PRINT"'#' suivi d'un nombre pour d
éplacer le pointeur"
17350 PRINT"'ç' pour quitter la fonction
"
17360 PRINT:INPUT"Votre choix: ",p$
17370 p$=UPPER$(p$):IF p$="" OR p$="C" T
HEN s1=s1+1
17380 IF p$="C" THEN GOTO 17420
17390 IF p$="A" THEN GOSUB 15000:p$=""
17400 IF LEFT$(p$,1)="#" THEN s1=s1+VAL(
MID$(p$,2)):p$=""
17410 WEND
17420 WEND
17430 RETURN
```

Commentaires

Ligne 17030 : S1, comme nous l'avons dit dans les commentaires sur le module précédent, est le pointeur sur l'entrée en cours; il repart à 0 à chaque début de recherche.

Lignes 17040-17060 : Un message d'erreur apparaît si aucune donnée n'a encore été placée dans le fichier.

Lignes 17070-17120 : C'est le menu initial du module de recherche. Pour chaque recherche, vous ne le verrez qu'une fois au début. Ce menu vous permet de spécifier le type de recherche; si vous voulez changer de type de recherche, vous devez quitter la recherche en cours et repasser par ce menu. Le terme

'RECHERCHE NORMALE' indique la recherche d'une combinaison donnée de caractères. La première entrée obtenue sera la première du fichier contenant ces caractères, dans n'importe quelle position. 'RECHERCHE INITIALE' indique la recherche d'une entrée *qui commence* avec la combinaison de caractères spécifiée par l'utilisateur. Ainsi, "SMI' trouverait une entrée commençant avec les lettres SMI, pas forcément la première. Si la chaîne spécifiée ne se trouve au début d'aucune entrée, l'entrée obtenue est celle qui se trouve à l'endroit où elle serait insérée s'il s'agissait d'une nouvelle entrée.

Vous pouvez étendre la recherche normale ou initiale sur plus d'une donnée d'une entrée, en plaçant un caractère ' ' dans la chaîne à rechercher. Si vous utilisez cette technique sur un fichier dont la première donnée de chaque entrée est un nom et la deuxième un prénom, une recherche initiale de 'DURAND' 'A' donnera tout 'DURAND' dont l'initiale est 'A', mais pas forcément le premier.

Dans une recherche normale, si la chaîne spécifiée n'est pas trouvée dans l'une des entrées du fichier, le programme revient au menu principal.

Ligne 17130 : Cette ligne fait tout le travail nécessaire à une 'recherche initiale', en supprimant l'astérisque de gauche et en appelant simplement le module de recherche binaire pour qu'il trouve la position correcte.

Lignes 17140-17420 : Cette boucle principale répètera une recherche normale si vous le demandez à partir d'un menu ultérieur. A noter que la routine de recherche initiale ne figure pas dans cette boucle car il est inutile de répéter une recherche initiale puisque la même entrée sera toujours obtenue.

Lignes 17150-17210 : A ce stade du déroulement du module, toute entrée doit être une chaîne faisant l'objet d'une recherche normale. Cela est fait simplement en analysant l'entrée avec INSTR. Si une donnée est trouvée, la variable FF reçoit la valeur 1. La position est enregistrée dans S1 puis la variable de la boucle, I, reçoit sa plus haute valeur afin que la boucle se termine.

Lignes 17220-17410 : Cette boucle continue tant que P$, l'entrée du menu de recherche suivant, représente une 'chaîne nulle'. La valeur de IT doit être incluse dans la condition de la boucle, parce que nous voulons pouvoir supprimer des données dans cette boucle. Si toutes les données sont supprimées, nous devrons sortir de la boucle, sous peine de créer une erreur.

Lignes 17230-17240 : Dans la boucle, vous pouvez vous déplacer dans le fichier à l'aide d'un nombre. Ces lignes contrôlent lors des passages suivants dans la boucle, que le pointeur n'a pas été déplacé au-delà des limites du nombre actuel d'entrées.

Lignes 17250-17290 : Après avoir appelé le module précédent, ces lignes utilisent l'information sur la position des caractères de séparation pour afficher les données constituant l'entrée en cours, avec le libellé de la donnée. A chaque passage dans la boucle FOR, les caractères situés entre la valeur de la variable PP et une valeur contenue dans le tableau PTR sont affichés. Lorsque chaque lot de caractères a été affiché, PP reçoit la valeur en cours dans le tableau PTR et la variable de la boucle, I, prend la valeur suivante dans PTR.

Lignes 17300-17360 : Ce menu apparaît lorsqu'une entrée est affichée. Vous avez plusieurs possibilités : passer à la donnée suivante, appeler la fonction 'M' (modifier) (pas encore entrée), continuer la recherche de la chaîne précédemment spécifiée, sauter dans le fichier d'un nombre d'entrées spécifié, ou revenir au menu du programme principal.

Lignes 17370-17380 : Ces deux lignes sont associées aux boucles commençant aux lignes 17140 et 17220. Si l'entrée est une 'chaîne nulle' (c'est-à-dire si vous appuyez sur ENTER), la boucle de la ligne 17220 est répétée et affiche l'entrée suivante indiquée par S1. Si vous entrez 'C', la boucle de la ligne 17140 exécute à nouveau la recherche en commençant à l'entrée suivante.

Ligne 17390 : La fonction M (modifier) qui n'a pas encore été entrée.

Ligne 17400 : L'entrée d'un nombre précédé du signe '#' vous permet de vous déplacer vers l'avant ou vers l'arrière dans le fichier. Le fait de donner une chaîne nulle à P$ exécute simplement la boucle de la ligne 17220 pour afficher l'entrée que vous venez d'atteindre.

Test

Si vous avez sauvegardé les suites de quatre entrées créées aux tests précédents, exécutez le programme et rechargez les entrées. Spécifiez l'option 4 sur le menu principal puis, lorsque le menu de recherche apparaît, faites défiler les entrées en appuyant sur ENTER. Chaque entrée doit s'afficher sur deux lignes, avec le nom de donnée approprié, par exemple :

UN :AA1
DEUX :AA2

A la quatrième entrée, vous constaterez que vous ne pouvez plus continuer en appuyant sur ENTER. Entrez '#-1' qui doit vous ramener à l'entrée 3.

Continuez à reculer ; vous constaterez que vous ne pouvez pas non plus aller au-delà du début du fichier.

Entrez ' ' pour revenir au menu principal et spécifiez l'option 4 à nouveau. Cette fois-ci répondez par 'CC' au menu de recherche initial. L'entrée 3 doit s'afficher, la seule qui contienne les caractères 'CC'. Vous êtes au deuxième menu de recherche, donc entrez 'C' pour continuer la recherche. Vous devez revenir au menu principal.

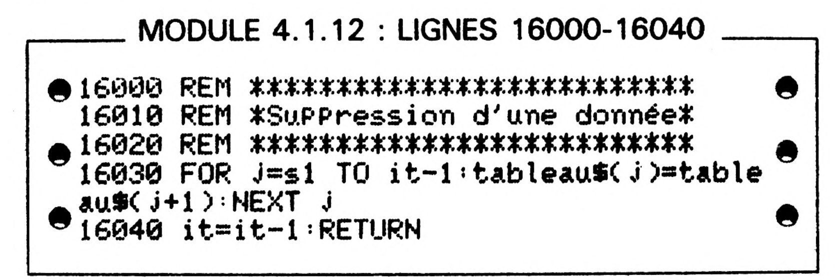
Choisissez à nouveau l'option 4 et entrez '2' comme cible de recherche. L'entrée 1 doit apparaître puisqu'elle contient le caractère '2'. Entrez 'C' pour continuer la recherche et l'entrée 2 doit s'afficher ; elle contient aussi le caractère '2'. Continuez à entrer 'C' jusqu'à ce que les quatre entrées aient été affichées et que la recherche échoue, vous ramenant au menu principal.

Pour finir, choisissez l'option 4 du menu principal et entrez "B' comme cible de recherche. L'entrée 2 doit s'afficher, la seule qui commence par 'B'. Entrez ' ' pour revenir au menu principal et mettez fin au programme.

Vous avez testé toutes les fonctions de recherche.

Module 4.1.12 : Suppression d'entrées

Les deux modules suivants qui permettent de modifier ou de supprimer les entrées, parachèvent le programme. Le module SUPPRESSION est ajouté en premier, puisqu'il est utilisé à chaque modification d'une entrée.



MODULE 4.1.12 : LIGNES 16000-16040

```
16000 REM ***************************
16010 REM *Suppression d'une donnée*
16020 REM ***************************
16030 FOR j=s1 TO it-1:tableau$(j)=table
au$(j+1):NEXT j
16040 it=it-1:RETURN
```

Commentaires

Lignes 16030-16040 : La suppression consiste simplement à

commencer à l'entrée au-dessus de celle qui doit être supprimée et à la copier une position plus bas. Lorsque chaque entrée a été déplacée, le comptage des données est réduit de 1.

Test

Exécutez le programme et lisez les quatre données sur cassette. Spécifiez l'option 4 pour arrêter le programme. Puis entrez :

S1=0[ENTER]
GOTO 16000

Le programme doit s'arrêter avec une erreur 'Unexpected RETURN' (RETURN inattendu). Entrez :

GOTO 11000[ENTER]

puis appelez l'option de recherche. Vous constaterez que l'entrée AA1 AA2 a disparu du fichier.

Module 4.1.13 : Modification des entrées

Pour que le programme soit utile, il est indispensable que nous puissions modifier les données existantes. C'est le rôle de ce module.

MODULE 4.1.13 : LIGNES 15000-15190

```
15000 REM ****************************
15010 REM *Modification d'une entrée*
15020 REM ****************************
15030 te$="":pp=0
15040 FOR i=0 TO x-1
15050 CLS:PRINT"Entrée ";s1+1;":"
15060 PRINT:PRINT item$(i);":";MID$(tabl
eau$(s1),pp+1,ptr(i)-pp-1)
15070 PRINT:PRINT"COMMANDES DISPONIBLES:
"
15080 PRINT:PEN 2:PRINT"ENTER";
15090 PEN 1:PRINT" laisse les données in
changées"
15100 PRINT"Entrer nouvelle donnée pour
remplacer celle affichée"
15110 PRINT"'?D' Supprime toute l'entrée
"
15120 PRINT"'?' laisse l'entrée inchangé
e"
15130 PRINT:INPUT"Que voulez-vous: ",q$
```

```
15140 q$=UPPER$(q$):IF q$="çD" THEN GOSU
B 16000:RETURN
15150 IF q$="ç" THEN RETURN
15160 IF q$<>"" THEN q$=q$+"ç"
15170 IF q$="" THEN q$=MID$(tableau$(s1)
,pp+1,ptr(i)-pp)
15180 pp=ptr(i):te$=te$+q$:NEXT i:GOSUB
16000:GOSUB 13000
15190 s1=ss:GOSUB 14000:RETURN
```

Commentaires

Lignes 15040-15180 : Cette boucle paraît semblable à celle qui affiche les entrées au module 4.1.8. Toutefois, elle n'affiche qu'une donnée à la fois.

Ligne 15140 : La frappe de ' D' lorsqu'une donnée est affichée, supprime toute l'entrée qui contient cette donnée. A noter qu'il n'est pas possible de supprimer une *donnée* individuelle puisque le nombre de données par entrée est fixe.

Ligne 15150 : La frappe de ' ' en réponse à toute donnée redonne la main au module de recherche. Les modifications éventuelles apportées aux données précédentes de l'entrée sont ignorées et l'entrée reste inchangée.

Ligne 15160 : Toute autre entrée que ' D' ou ' ', est considérée comme un remplacement de la donnée affichée. Le séparateur ' ' est ajouté à la fin de la donnée, comme dans le module de nouvelles données.

Ligne 15170 : Si vous appuyez sur ENTER sans autre entrée, la donnée affichée est copiée sans modification. Pour ne modifier qu'une donnée, appuyez simplement sur ENTER pour toutes les autres. Notez la différence entre cette expression chaîne et celle qui a permis d'afficher la donnée à la ligne 15060. Le '-1' à la fin est abandonné afin que le caractère de séparation ' ' ne soit pas supprimé.

Lignes 15180-15190 : L'entrée modifiée est reconstituée dans TE$. Lorsqu'elle est terminée, l'entrée initiale est supprimée du fichier. La raison en est que les modifications apportées ont peut-être modifié la position correcte de l'entrée dans le fichier. Après sa suppression, l'entrée est envoyée au module de recherche binaire puis réinsérée, sa position dans le fichier ayant été copiée dans la variable S1 ; ainsi, le module de recherche sait quelle donnée afficher si la position a été modifiée.

Test

Exécutez le programme et lisez les quatre données sur cassette. Choisissez l'option de recherche et appuyez sur ENTER pour afficher l'entrée 1. Puis entrez 'M' en réponse au deuxième menu de recherche. La première donnée, 'AA1', doit s'afficher avec le menu de modification (AMEND). Entrez 'AAA1' et, lorsque la deuxième donnée est affichée, appuyez sur ENTER. Vous devez vous retrouver au module de recherche et l'entrée doit être affichée sous la forme suivante :

UN :AAA1
DEUX :AA2

Essayez d'apporter d'autres modifications et de supprimer des entrées.

PROGRAMME 4.2 : NNOMBRE

Fonction du programme

Le classement et les fichiers ne concernent pas que des mots. Les micro- ordinateurs sont aussi parfaitement capables de stocker et de traiter des nombres. Ce programme 'NNombre' (contraction de 'Nom et Nombre') vous permet de stocker des noms d'articles, leur unité habituelle de mesure et une quantité associée. Avant de déclarer que vous ne voyez pas l'intérêt d'un tel programme, mettez-vous à la place d'un petit commerçant ou même d'une cuisinière.

Le commerçant a une masse d'articles qui constituent son stock. Tous les articles du stock ont des noms, et se présentent sous des unités différentes (boîte, bouteille, sac, etc.) et tous ont une donnée très importante qui leur est associée : le prix. Donc, pour qu'un micro-ordinateur puisse gérer le stock ou établir une facture, il doit connaître ces trois données pour chaque article. A la maison, les aliments que nous consommons ont tous un nom, leurs unités sont également différentes (cuillerée, grammes, décilitre, etc.) et, si nous nous préoccupons de notre ligne, il faut tenir compte d'une autre valeur : les 'calories'.

Il ne s'agit là que d'exemples et vous pouvez en imaginer bien d'autres où il est important d'enregistrer des noms, des unités et une quantité associée à toute une variété d'articles.

Nnombre vous permet de créer un 'dictionnaire' d'articles - jusqu'à 200 - avec leurs unités de mesure et les valeurs associées à ces unités. A partir de ce dictionnaire, vous pourrez construire des listes d'articles que le programme affichera et totalisera. Nnombre est facile d'emploi, qu'il s'agisse d'ajouter les calories des recettes d'un jour ou de donner le prix total d'un ensemble de marchandises.

Module 4.2.1 : Initialisation

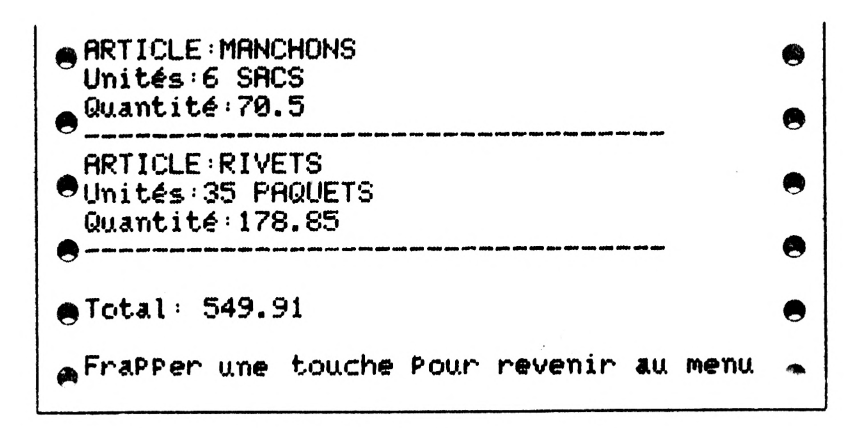
Il s'agit d'un module standard. Vous devez simplement spécifier le type d'articles que le programme va traiter. Ce sera un nom du type 'Denrée comestible' ou 'Article de stock'.

MODULE 4.2.1 : LIGNES 10000-10090

```
10000 REM ****************
10010 REM *Initialisation*
10020 REM ****************
10030 MODE 1
10040 DIM tableau$(1000,1),tableau(1000)
,te$(100,1),te(100) : cu=0 : it=0
10050 PEN 2:PRINT TAB(17);"NNOMBRE":PRIN
T TAB(17);"======="
10060 WINDOW 1,40,4,25
10070 PEN 1
10080 INPUT "Lecture sur cassette (o/n)"
;q$
10090 IF LOWER$(q$)="o" THEN GOSUB 22000
 ELSE PRINT:INPUT "Nom général des artic
les:",nn$
```

```
                NNOMBRE
                =======

ARTICLE:RONDELLES
Unités:3 BOITES
Quantité:8.07
------------------------------------
ARTICLE:BRIDES
Unités:11 BOITES
Quantité:292.49
------------------------------------
```



```
ARTICLE:MANCHONS
Unités:6 SACS
Quantité:70.5
-----------------------------------
ARTICLE:RIVETS
Unités:35 PAQUETS
Quantité:178.85
-----------------------------------

Total: 549.91

Frapper une touche pour revenir au menu
```

Figure 4.2 : Nnombre en mode liste en cours.

Commentaires

Ligne 10040 : Le tableau TABLEAU$ est utilisé pour enregistrer le nom d'article et le nom d'unité de chacun des articles du dictionnaire. La quantité associée pour chaque unité est stockée dans l'élément équivalent du tableau TABLEAU.

TE$ et TE jouent le même rôle que TABLEAU$ et TABLEAU, mais pour la 'liste en cours' extraite du dictionnaire. La variable CU enregistre le nombre d'articles dans la 'liste en cours' : la liste dérivée du dictionnaire principal. IT, comme dans la plupart des programmes de ce livre, enregistre le nombre d'articles dans le fichier principal, dans ce cas le dictionnaire.

Module 4.2.2 : Menu

Il s'agit d'un module de menu standard. Seule différence avec les programmes précédents : la façon dont la ligne 11160 empêche d'accéder à certaines fonctions du programme avant que des données n'aient été entrées ; cette fonction ne peut pas être utilisée avant l'entrée du module suivant.

MODULE 4.2.2 : LIGNES 11000-11210

```
11000 REM ****************
11010 REM *Menu Principal*
11020 REM ****************
11030 WHILE z<>8
11040 CLS
```

```
11050 PRINT"COMMANDES DISPONIBLES:"
11060 PRINT
11070 PRINT "1) Afficher la liste en cou
rs"
11080 PRINT "2) Entrer dans la liste en
cours"
11090 PRINT "3) Commencer une nouvelle l
iste"
11100 PRINT "4) Supprimer la liste en co
urs"
11110 PRINT "5) Ajouter au dictionnaire"
11120 PRINT "6) Examiner les articles du
 dictionnaire"
11130 PRINT "7) Sauvegarder les articles
sur cassette"
11140 PRINT "8) Arreter"
11150 PRINT:INPUT"Votre choix:";z
11160 IF (z=1 OR z=2 OR z=4 OR z=6 OR z=
7) AND it=0 THEN x$="********* PAS ENCOR
E D'ENTREE *********":GOSUB 23000:z=0
11170 ON z GOSUB 12000,13000,14000,20000
,15000,18000,21000
11180 WEND
11190 CLS
11200 LOCATE 11,11:PRINT "PROGRAMME TERM
INE"
11210 END
```

Module 4.2.3 : Messages d'erreur

Ce petit module peut épargner de la mémoire dans le cas d'un programme qui génère un certain nombre de messages d'erreur différents. Ce module ne fait rien d'autre qu'afficher une chaîne nommée X$, d'émettre un signal sonore et de revenir à sa place. Vous pouvez voir à la ligne 11160 que, pour appeler le module, X$ est défini puis une commande GOSUB est émise.

MODULE 4.2.3 : LIGNES 23000-23070

```
23000 REM ********
23010 REM *Erreur*
23020 REM ********
23030 PRINT:PRINT x$
23040 SOUND 1,1000,100
```

```
23050 FOR i=1 TO 1500
23060 NEXT i
23070 RETURN
```

Test

Exécutez le programme et répondez aux messages de façon appropriée. Lorsque le menu apparaît, spécifiez l'option 1 qui doit afficher le message d'erreur indiquant qu'il n'y a pas encore de données en mémoire.

Modules 4.2.4 et 4.2.5 : Fichiers de données

Deux modules standard.

MODULES 4.2.4 et 4.2.5 : LIGNES 21000-22210

```
21000 REM **************************
21010 REM *Sauvegarde sur cassette*
21020 REM **************************
21030 CLS
21040 PRINT "Sauvegarder les données:":P
RINT
21050 INPUT "Nom de fichier:",fichier$
21060 OPENOUT fichier$
21070 PRINT #9,nn$
21080 PRINT #9,cu
21090 PRINT #9,it
21100 FOR i=0 TO cu-1
21110 PRINT #9,te$(i,0)
21120 PRINT #9,te$(i,1)
21130 PRINT #9,te(i)
21140 NEXT i
21150 FOR i=0 TO it-1
21160 PRINT #9,tableau$(i,0)
21170 PRINT #9,tableau$(i,1)
21180 PRINT #9,tableau(i)
21190 NEXT i
21200 CLOSEOUT
21210 RETURN
22000 REM ***********************
22010 REM *Lecture sur cassette*
22020 REM ***********************
```

```
22030 CLS
22040 PRINT "Charger les données:":PRINT
22050 INPUT "Nom de fichier:",fichier$
22060 OPENIN fichier$
22070 INPUT #9,nn$
22080 INPUT #9,cu
22090 INPUT #9,it
22100 FOR i=0 TO cu-1
22110 INPUT #9,te$(i,0)
22120 INPUT #9,te$(i,1)
22130 INPUT #9,te(i)
22140 NEXT i
22150 FOR i=0 TO it-1
22160 INPUT #9,tableau$(i,0)
22170 INPUT #9,tableau$(i,1)
22180 INPUT #9,tableau(i)
22190 NEXT i
22200 CLOSEIN
22210 RETURN
```

Module 4.2.6 : Recherche binaire

Pour un commentaire complet de ce module, reportez-vous au module équivalent d'Unifile. Notons simplement que le classement est fait sur la base du nom d'article dans la colonne 0 de TABLEAU$.

MODULE 4.2.6 : LIGNES 16000-16110

```
16000 REM *******************
16010 REM *Recherche Binaire*
16020 REM *******************
16030 IF it=0 THEN ss=0:RETURN
16040 Po=INT(LOG(it)/LOG(2)):ss=2^Po-1
16050 FOR i=Po TO 0 STEP -1
16060 ss=ss+2^i*((tableau$(ss,0)>t1$)-(t
ableau$(ss,0)<t1$))
16070 IF ss<0 THEN ss=0
16080 IF ss>it-1 THEN ss=it-1
16090 NEXT i
16100 IF tableau$(ss,0)<t1$ THEN ss=ss+1
16110 RETURN
```

Module 4.2.7 : Insertion d'articles dans le dictionnaire principal

Le principe de ce module est identique à son homologue d'Unifile. Il est ici un peu plus long car il doit insérer deux chaînes et un nombre dans les deux tableaux au lieu d'une seule chaîne.

MODULE 4.2.7 : LIGNES 17000-17110

```
17000 REM ********************
17010 REM *Insertion d'article*
17020 REM ********************
17030 FOR i=it TO ss+1 STEP -1
17040 tableau$(i,0)=tableau$(i-1,0)
17050 tableau$(i,1)=tableau$(i-1,1)
17060 tableau(i)=tableau(i-1)
17070 NEXT i
17080 tableau$(ss,0)=t1$
17090 tableau$(ss,1)=t2$
17100 tableau(ss)=nn
17110 RETURN
```

Module 4.2.8 : Entrée d'articles pour le dictionnaire

Ce module est bien moins compliqué que son homologue d'Unifile; il accepte trois entrées : (a) le nom de l'article, (b) le nom des unités de mesure de l'article et (c) la quantité associée à ces unités.

MODULE 4.2.8 : LIGNES 15000-15170

```
15000 REM ***************************
15010 REM *Extention du Dictionnaire*
15020 REM ***************************
15030 WHILE 1
15040 CLS
15050 PRINT "Nouveaux articles pour dict
ionnaire:":PRINT
15060 IF it>1000 THEN x$="    *********
PLUS DE PLACE *********":GOSUB 23000:RET
URN
15070 q$="":WHILE LOWER$(q$)<>"o"
15080 PRINT nn$;:INPUT " (nom ou '?' pou
```

```
r quitter):",t1$
15090 IF t1$="ç" THEN RETURN
15100 PRINT "Unités";:INPUT ":",t2$
15110 PRINT "Quantité pour ";LOWER$(t2$)
;:INPUT ":",nn
15120 PRINT:INPUT "Sont-ils corrects ?(o
/n)";q$:PRINT
15130 WEND
15140 GOSUB 16000
15150 GOSUB 17000
15160 it=it+1
15170 WEND
```

Exécutez le programme, spécifiez que vous ne lisez pas à partir d'une cassette et donnez le nom ARTICLE en réponse au message demandant un nom général. Sur le menu principal, choisissez l'option 5 'Ajouter au dictionnaire'. Lorsque l'écran 'nouveaux articles' apparaît, entrez les trois lignes suivantes :

OBJET1/BOITE/10
OBJET2/BOUTEILLE/20
OBJET3/SAC/40

Ces articles n'ont pas de signification particulière et ne servent qu'au test. Après avoir entré les articles, revenez au menu principal en entrant ' '. Puis appelez le module du fichier de données (option 7) pour stocker l'information. Arrêtez le programme avec l'option 8 et entrez :

FOR I=0 TO 2 :?TABLEAU$(I,0),TABLEAU$(I,1),TABLEAU(I) :NEXT [ENTER]

Vous devez obtenir :

OBJET 1	BOITE	10
OBJET 2	BOUTEILLE	20
OBJET 3	SAC	40

A présent, exécutez le programme en spécifiant que vous *voulez* lire sur cassette. Tapez le nom donné lors du stockage. Après la lecture de la cassette, vous devez pouvoir effectuer le même test sur le contenu des tableaux, avec le même résultat.

Test

Vous pouvez enfin effectuer un test réel sur ce qui a été entré jusque-là.

Module 4.2.9 : La routine de recherche

Comme avec Unifile, ce module vous permet d'explorer le fichier des articles du dictionnaire, de rechercher certains articles ou d'en supprimer du fichier. Ce module est plus simple que celui d'Unifile, car il est conçu pour rechercher des articles entiers, et non des combinaisons de caractères stockés n'importe où dans un article. En outre, la structure d'une entrée complète de Nnombre est bien plus simple que la structure d'une donnée du tableau principal d'Unifile.

MODULE 4.2.9 : LIGNES 18000-18250

```
18000 REM ***********
18010 REM *Recherche*
18020 REM ***********
18030 ss=0
18040 t1$="":WHILE it>0
18050 CLS:PRINT "Recherche:":PRINT
18060 PRINT "Numéro de l'article:";ss+1
18070 PRINT nn$;":";tableau$(ss,0)
18080 PRINT "Unités:";tableau$(ss,1)
18090 PRINT "Quantité pour ";LOWER$(tabl
eau$(ss,1));":";tableau(ss)
18100 PRINT:PRINT "Commandes disponibles
:":PRINT
18110 PRINT "Entrer l'article à recherch
er"
18120 PEN 2:PRINT "ENTER";:PEN 1:PRINT "
 pour l'article suivant"
18130 PRINT "'#' suivi d'un nombre pour
déplacer le pointeur"
18140 PRINT "'s' pour supprimer l'articl
e"
18150 PRINT "'ç' pour quitter"
18160 PRINT:INPUT "Que voulez-vous";t1$
18170 IF t1$="" THEN t1$="#1"
18180 IF t1$="ç" THEN RETURN
18190 IF LOWER$(t1$)="d" THEN GOSUB 1900
0:t1$=""
18200 IF LEFT$(t1$,1)="#" THEN ss=ss+VAL
(MID$(t1$,2)):t1$=""
18210 IF t1$<>"" THEN GOSUB 16000
18220 IF ss>it-1 THEN ss=it-1
18230 IF ss<0 THEN ss=0
18240 WEND
```

```
18250 RETURN
18210 IF t1$<>"" THEN GOSUB 16000
18220 IF ss>it-1 THEN ss=it-1
18230 IF ss<0 THEN ss=0
18240 WEND
18250 RETURN
```

Commentaires

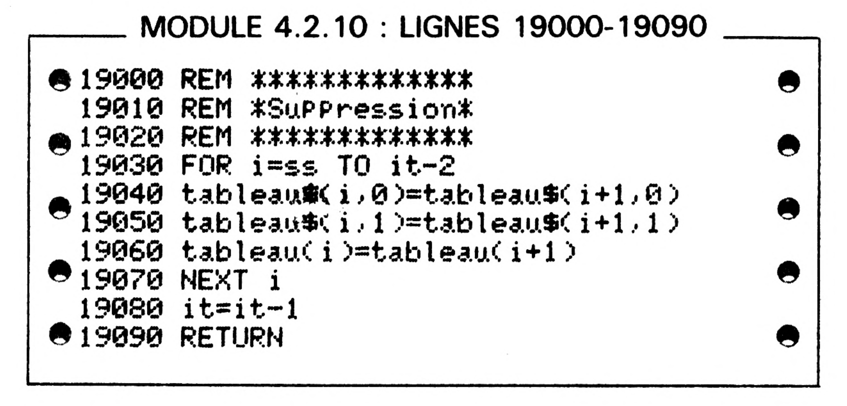
Ligne 18170 : Plutôt que de prévoir l'entrée de RETURN - c'est-à-dire une chaîne vide - la variable T1$ reçoit d'abord la valeur #1. Si vous appuyez simplement sur RETURN, le contenu de T1$ reste inchangé et l'article suivant du fichier s'affiche.

Test

Exécutez le programme, lisez les trois articles sur cassette, puis choisissez l'option 6, 'Examiner les données du dictionnaire' du menu principal. Vous pouvez consulter les articles, vers l'avant ou vers l'arrière, en utilisant un nombre précédé d'un '#' comme dans Unifile. Vous pouvez aussi obtenir un article en entrant son nom, mais pas le nom de l'unité.

Module 4.2.10 : Suppression d'un article

Equivalent direct du module de suppression d'Unifile.



MODULE 4.2.10 : LIGNES 19000-19090

```
19000 REM ************
19010 REM *Suppression*
19020 REM ************
19030 FOR i=ss TO it-2
19040 tableau$(i,0)=tableau$(i+1,0)
19050 tableau$(i,1)=tableau$(i+1,1)
19060 tableau(i)=tableau(i+1)
19070 NEXT i
19080 it=it-1
19090 RETURN
```

Test

Exécutez le programme, lisez les données sur cassette, choisissez l'option 6 du menu principal, et tapez 'S' pour l'une des entrées. Vous constaterez que l'entrée est supprimée du fichier.

Module 4.2.11 : Copie d'articles dans la liste en cours

Nnombre n'a pas seulement pour but de gérer un dictionnaire d'articles et des quantités associées, mais aussi d'utiliser ce dictionnaire comme base de construction de listes temporaires.

Les modules suivants sont donc conçus pour vous permettre d'ajouter des articles à la liste 'en cours', d'afficher cette liste, d'en supprimer certains articles ou de supprimer toute la liste en une seule opération. Ce module permet de copier les articles du dictionnaire principal dans la liste 'en cours'.

MODULE 4.2.11 : LIGNES 13000-13210

```
13000 REM ***********************
13010 REM *Extension de la liste*
13015 REM *      en cours       *
13020 REM ***********************
13030 WHILE 1
13040 CLS
13050 PRINT "Ajouter à la liste en cours
:":PRINT
13060 IF cu>100 THEN x$="    ***** LISTE
 EN COURS PLEINE *****":GOSUB 22000:RETU
RN
13070 PRINT nn$;:INPUT " ('ç' pour quitt
er):",t1$
13080 IF t1$="ç" THEN RETURN
13090 connu=0:GOSUB 16000:IF tableau$(ss
,0)=t1$ THEN connu=1
13100 IF connu=0 THEN x$="**** "+UPPER$(
nn$)+" INCONNU, VERIFIER ****":GOSUB 230
00:RETURN
13110 PRINT:PRINT "Unités:";tableau$(ss,
1)
13120 INPUT "Quantité:",q
13130 PRINT: INPUT "Sont-ils corrects (o
/n)";q$
13140 WHILE LOWER$(q$)="o"
13150 te$(cu,0)=tableau$(ss,0)
13160 te$(cu,1)=MID$(STR$(q),2)+" "+tabl
eau$(ss,1)
13170 te(cu)=q*tableau(ss)
13180 cu=cu+1
13190 q$=""
```

```
13200 WEND
13210 WEND
```

Commentaires

Lignes 13090-13100 : Ces lignes vérifient si l'article entré par l'utilisateur pour aller dans la liste en cours, est présent dans le dictionnaire principal. Pour cela, le module de recherche binaire est appelé pour obtenir la position à laquelle serait inséré l'article dans le dictionnaire. Puis le contenu réel du dictionnaire est comparé à l'entrée. Si l'entrée de l'utilisateur est dans le dictionnaire, les deux articles sont identiques ; sinon un message d'erreur apparaît et le module se termine.

Ligne 13110 : Après avoir trouvé l'article dans le dictionnaire, le module affiche les unités de mesure normales et vous demande combien de ces unités il faut inclure.

Lignes 13120-13170 : Entrez les valeurs réelles. Puis vous devez confirmer l'exactitude de l'entrée avant qu'elle ne s'ajoute à la liste en cours, contenue dans les variables TE$ et TE. Notez que la quantité stockée en TE n'est pas la quantité par unité provenant du dictionnaire, mais la quantité *totale* du nombre d'unités que vous avez spécifiées.

Test

Exécutez le programme et lisez les données sur cassette. Choisissez l'option 2 du menu principal. Entrez les articles et les nombres d'unités suivants :

OBJET1,1
OBJET2,2
OBJET3,3

Essayez ensuite de faire accepter au module, 'OBJET4', qui n'est pas présent dans le dictionnaire. Un message d'erreur va vous demander de vérifier le nom de l'article. Avant toute chose, choisissez l'option 7 afin de stocker la liste en cours récemment créée ainsi que le dictionnaire principal. Pour finir, choisissez l'option 8 pour arrêter le programme.
Puis entrez la ligne suivante en mode direct :
FOR I=0 TO 2 : ?TE$(I,0),TE0$(I,1),TE(I) :NEXT[ENTER]
Vous devez obtenir :

OBJET1	1 BOITE	10
OBJET2	2 BOUTEILLE	40
OBJET3	3 SAC	120

Module 4.2.12 : Affichage de la liste en cours

Ce module a pour seul rôle d'afficher une à une les entrées qui constituent la liste en cours. Après chaque entrée, vous êtes invité à frapper une touche pour afficher l'entrée suivante. Cela, parce que la liste est normalement plus longue que l'écran et qu'elle ne doit pas défiler trop rapidement. A la fin de la liste, le total des quantités correspondant au contenu de la liste en cours est fourni.

MODULE 4.2.12 : LIGNES 12000-12160

```
12000 REM **********************
12010 REM *Affichage de la liste*
12015 REM *       en cours       *
12020 REM **********************
12030 IF cu=0 THEN RETURN
12040 CLS:ct=0
12050 FOR i=0 TO cu-1
12060 PRINT nn$;":";te$(i,0)
12070 PRINT "Unités:";te$(i,1)
12080 PRINT "Quantité:";te(i)
12090 PRINT "-----------------------------
-----------"
12100 WHILE INKEY$="":WEND
12110 ct=ct+te(i)
12120 NEXT i
12130 PRINT:PRINT "Total:";ct
12140 PRINT:PRINT "Frapper une touche po
ur revenir au menu"
12150 WHILE INKEY$="":WEND
12160 RETURN
```

Module 4.2.13 : Suppression d'articles de la liste en cours

Il s'agit d'une version très simplifiée du module de recherche utilisé pour le dictionnaire principal. Il vous permet de consulter la liste en cours, un article à la fois, et de supprimer l'article de votre choix en tapant 'S' en face. Vous pouvez quitter cette opération à tout moment en entrant ' '.

MODULE 4.2.13 : LIGNES 20000-20050

```
20000 REM ************************
20010 REM *Suppression de la liste*
20015 REM *       en cours        *
20020 REM ************************
20030 i=0
20040 WHILE i<cu
20050 CLS
20060 PRINT "Supprimer la liste en cours
:":PRINT
20070 PRINT te$(i,0)
20080 PRINT te$(i,1)
20090 PRINT:PRINT "Commandes disponibles
:":PRINT
20100 PEN 2:PRINT "ENTER";:PEN 1:PRINT "
pour l'article suivant"
20110 PRINT "'d' pour supprimer"
20120 PRINT "'ç' pour quitter"
20130 PRINT:INPUT "Que voulez-vous";q$
20140 IF q$="ç" THEN RETURN
20150 IF LOWER$(q$)<>"d" THEN i=i+1
20160 WHILE LOWER$(q$)="d"
20170 FOR j=i TO cu-1
20180 te$(j,0)=te$(j+1,0)
20190 te$(j,1)=te$(j+1,1)
20200 te(j)=te(j+1)
20210 NEXT j
20220 cu=cu-1
20230 q$="":WEND
20240 WEND
20250 RETURN
```

Commentaires

Lignes 20150-20230 : La variable I contrôle quels éléments sont affichés. A noter qu'elle n'est *pas* incrémentée si vous entrez 'S'. I va pointer sur l'article à supprimer et l'opération de suppression va copier l'article inférieur suivant sur cette position.

Test

Exécutez le programme et rechargez le fichier contenant la liste en cours, sauvegardée lors d'un test précédent. Choisissez l'option 4, 'Supprimer de la liste en cours', du menu principal.

Vous pouvez consulter les trois articles de la liste en cours et en supprimer un ; vérifiez sa suppression, en affichant la liste en cours.

Module 4.2.14 : Initialisation de la liste en cours

Dans de nombreux cas, il faudra construire une liste en cours, obtenir le total correspondant, puis passer rapidement à une nouvelle liste. Ce module efface le contenu de la liste en cours en une seule opération.

MODULE 4.2.14 : LIGNES 14000-14090

```
14000 REM ****************************
14010 REM *Initialisation de la liste*
14015 REM *          en cours        *
14020 REM ****************************
14030 FOR i=0 TO cu-1
14040 te$(i,0)=""
14050 te$(i,1)=""
14060 te(i)=0
14070 NEXT i
14080 cu=0
14090 RETURN
```

Test

Exécutez le programme, rechargez le fichier qui contient la liste en cours et spécifiez l'option 3, 'Commencer une nouvelle liste', sur le menu principal. Le menu principal doit réapparaître presque aussitôt. Essayez ensuite de sélectionner l'option 1 du menu principal. Là encore, le menu principal réapparaît ; en effet, le module d'affichage a été appelé mais, comme il n'y a rien à afficher, l'exécution reprend immédiatement.

Si ce test est totalement satisfaisant, le programme est opérationnel.

PROGRAMME 4.3 : TEXTE

Fonction du programme

Le traitement de texte constitue l'une des applications les plus séduisantes des ordinateurs modernes. Avec un système de traitement de texte élaboré, tout le côté fastidieux lié à la rédaction quelle qu'elle soit, disparaît et les documents les plus complexes se transforment en tâches relativement simples. Malheureusement, un logiciel de traitement de texte complet peut nécessiter plus de mémoire que n'en possède un micro même plus puissant que le 464. De plus, quelle que soit la rapidité du BASIC du 464, la vitesse exigée par un logiciel de traitement de texte demande normalement une programmation en code machine, le langage mystérieux compris par le processeur Z80 du 464.

Le programme texte qui va suivre, n'a pas la prétention d'être un logiciel de traitement de texte. Toutefois, il vous permet d'utiliser votre 464 comme une machine à écrire beaucoup plus souple. Vous pourrez créer un texte, le modifier, lui ajouter des lignes ou des mots, déplacer des lignes et l'imprimer sur une imprimante compatible. Pour une utilisation professionnelle, ce n'est pas l'idéal, mais pour de petits documents, Texte convient parfaitement. Sachez que l'une de mes plus grandes satisfactions en tant qu'auteur est de recevoir des lettres de lecteurs des derniers livres, indiquant qu'ils ont rédigé leur lettre en utilisant les versions précédentes de Texte.

```
                          123 rue Pimpon
                          Bat. 3
                          Vermenon
                          Tel. 42 01 02 03

Cher Monsieur Blanc,

Merci pour votre lettre concernant le développement de Texte,
en remplacement de Wordstar. Je pense que nous n'en sommes
pas encore là.

Comme vous le voyez, les possibilités sont plutôt limitées.
Toutefois, il fait bien son travail.
Dans l'attente de votre prochain courrier,

Je reste votre dévoué

David Laurent
```

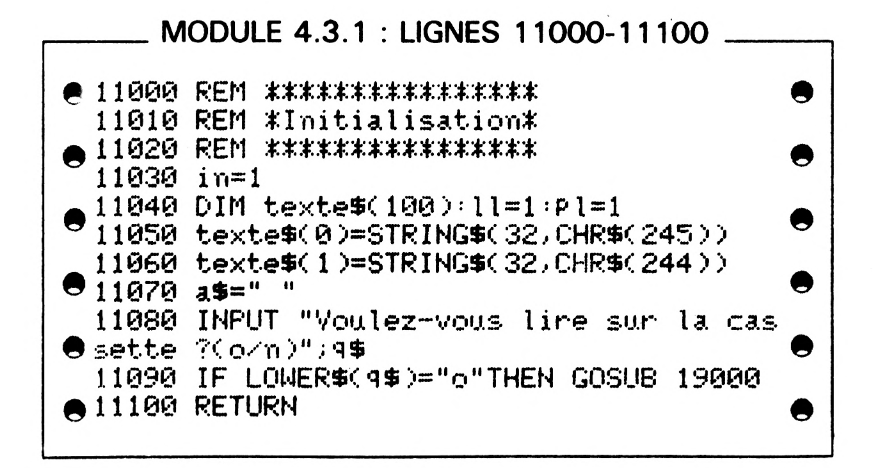
Figure 4.3 : Exemple de lettre composée et imprimée avec Texte

Voici les nouveaux concepts présentés dans ce programme :

1) Curseurs contrôlés par programme et entrée de texte.
2) Manipulation de données sous forme de grands tableaux de chaînes.
3) Sortie d'un texte complexe sur imprimante.

Module 4.3.1 : Initialisation

Il s'agit d'un module simple sans rien de particulier, si ce n'est l'usage d'une ou deux des variables déclarées.



MODULE 4.3.1 : LIGNES 11000-11100

```
11000 REM ***************
11010 REM *Initialisation*
11020 REM ***************
11030 in=1
11040 DIM texte$(100):ll=1:Pl=1
11050 texte$(0)=STRING$(32,CHR$(245))
11060 texte$(1)=STRING$(32,CHR$(244))
11070 a$=" "
11080 INPUT "Voulez-vous lire sur la cas
sette ?(o/n)";q$
11090 IF LOWER$(q$)="o"THEN GOSUB 19000
11100 RETURN
```

Commentaires

Ligne 11040 : Le tableau TEXTE$ va contenir la partie principale du texte entré.

Lignes 11050-11060 : Ces deux lignes de symboles graphiques créent deux marques qui apparaîtront sur l'écran lorsque vous rencontrerez la limite supérieure ou inférieure du bloc de texte.

Module 4.3.2 : Entrée des caractères

La tâche principale du programme consiste à accepter l'entrée au clavier et à en *faire* quelque chose. C'est le rôle de ce module. La plupart des lignes du module traitent l'entrée de caractères de 'commande' spéciaux qui demandent à Texte d'effectuer une fonction spécifique, par exemple supprimer un caractère. Certaines lignes ressemblent beaucoup au chapitre sur le graphisme, lorsqu'il faut un curseur mobile.

Ce module a pour principale fonction de vous permettre d'entrer du texte sur la liste 23 de l'écran, en l'éditant au fur et à mesure. Des modules ultérieurs prendront ces lignes et les intégreront au bloc principal de texte.

Notez également qu'il s'agit de l'un des rares programmes de l'ouvrage qui présente des séparations entre les lignes, pour plus de clarté. En effet, bien que relativement court, le programme utilise une telle masse de noms de variables et de petites boucles, qu'il est très difficile d'en suivre le déroulement sans une aide visuelle. Comme pour les programmes précédents ainsi présentés, les lignes ne contenant que le signe deux-points (:) peuvent être éliminées si vous le souhaitez.

MODULE 4.3.2 : LIGNES 12000-12430

```
12000 REM ****************
12010 REM *Edition de ligne*
12020 REM ****************
12030 :
12040 :
12050 WHILE 1
12060 :
12070 :
12080 t$="":WHILE t$=""
12090 LOCATE p-40*INT(p/40)+1,23+INT(p/4
0):PEN 1:PRINT CHR$(143)
12100 FOR tt=1 TO 20:NEXT tt
12110 LOCATE p-40*INT(p/40)+1,23+INT(p/4
0):PEN 2:PRINT MID$(a$,p+1,1)
12120 t$=INKEY$
12130 WEND
12140 :
12150 :
12160 IF t$=CHR$(13) OR (LEN(a$)>40 AND
t$=" ") THEN t$="":GOSUB 13000
12170 IF t$="^" THEN GOSUB 15000
12180 :
12190 :
12200 WHILE t$="ç"
12210 IF p<>0 THEN t=0 ELSE t=LEN(a$)-1
12220 p=t
12230 t$="":WEND
12240 :
12250 :
12260 WHILE p>0 AND t$=CHR$(127)
```

```
12270 a$=LEFT$(a$,p-1)+MID$(a$,p+1)
12280 p=p-1
12290 t$="":WEND
12300 :
12310 :
12320 WHILE t$>=CHR$(32) AND t$<=CHR$(16
3) AND t$<>CHR$(127)
12330 a$=LEFT$(a$,p)+t$+MID$(a$,p+1)
12340 p=p+1
12350 t$="":WEND
12360 :
12370 :
12380 IF t$=CHR$(242) AND p>0 THEN p=p-1
12390 IF t$=CHR$(243) AND p<LEN(a$)-1 TH
EN p=p+1
12400 :
12410 :
12420 LOCATE 1,23:PRINT a$
12430 WEND
```

Commentaires

Lignes 12080-12130 : Un module de clignotement du curseur qui fait clignoter un espace inverse sur un caractère sur la ligne 23. La valeur de la variable P indique la position du curseur. Le curseur alterne avec une lettre de la chaîne A$, utilisée pour entrer du texte.

Ligne 12160 : Cette ligne appelle la partie suivante du programme qui intègre l'entrée dans le bloc principal de texte. Cette intégration peut se faire de deux façons : soit en appuyant sur ENTER soit, automatiquement, dès que la longueur de texte dépasse 40 caractères et qu'un espace est entré.

Ligne 12170 : Le fait de frapper la touche portant le symbole 'flèche vers le haut' sur la ligne supérieure des touches appelle un module suivant qui permet diverses fonctions d'édition sur le bloc principal de texte.

Lignes 12200-12230 : Le fait de frapper ' ' déplace le curseur clignotant au début de la ligne ou, s'il y est déjà, à la fin de la ligne.

Lignes 12260-12290 : Pourvu que le curseur clignotant ne soit pas sur le premier caractère, la touche DEL supprime le caractère placé à gauche du curseur. Cela est facilement accompli en reconstruisant la chaîne A$ sans le caractère en position P.

Lignes 12320-12350 : Ces lignes acceptent tout caractère dont le code est compris entre 32 et 163 (voir la liste complète en annexe), *sauf* la touche DEL traitée précédemment, et l'intègrent à la ligne de texte entrée. Le nouveau caractère est inséré dans le texte à la position du curseur clignotant.

Lignes 12380-12390 : Pour déplacer le curseur clignotant vers la droite ou la gauche, il suffit de changer la valeur de P en fonction de la touche de déplacement du curseur actionnée. A noter que, comme au chapitre sur le graphisme, le fait de définir notre propre curseur nous permet de n'accepter que ce que nous voulons des touches de contrôle, telles que les touches fléchées, insert, delete et autres. Aucune de ces touches n'a d'effet automatique car elles ne sont pas comprises dans les caractères normaux d'impression que la boucle précédente accepte. Toutefois, nous pouvons les utiliser presque comme des touches de fonction en définissant leur effet ; cela est vrai pour toutes, excepté la touche ESC, bien entendu, qui arrête toujours le programme.

Test

Comme aux programmes précédents, vous devez commencer par insérer des lignes de la boucle de contrôle. Donc entrez les lignes suivantes :

```
10040 IF IN=0 THEN GOSUB 11000
10060 GOSUB 12000
```

puis exécutez le programme. Indiquez que vous ne voulez pas la lecture sur cassette et vous devez obtenir un curseur clignotant au début d'une ligne sur la moitié inférieure de l'écran. Ensuite commencez à taper : votre frappe doit s'afficher. Vous pouvez supprimer des caractères en utilisant la touche DEL, déplacer le curseur graphique sur ce que vous venez de taper ou le faire sauter du début à la fin en utilisant la touche ' '. La touche ENTER, les touches fléchées de contrôle ou le fait d'entrer plus de deux lignes de caractères suivies d'un espace, mettent fin au programme avec un message d'erreur 'Undefined line' (Ligne indéfinie).

Module 4.3.3 : Insertion de données dans le bloc principal de texte

Les deux modules suivants fonctionnent conjointement. Ils traitent les lignes entrées au bas de l'écran et les intègrent dans le bloc principal de texte contenu dans TEXT$. Ce bloc de texte peut contenir jusqu'à 100 lignes de 32 caractères. Le

travail proprement dit est fait par le présent module, mais les résultats ne seront apparents qu'après l'entrée du module suivant qui affiche un segment du texte.

MODULE 4.3.3 : LIGNES 13000-13310

```
13000 REM *******************
13010 REM *Insertion de ligne*
13020 REM *******************
13030 IF a$="* " THEN a$=SPACE$(33)
13040 x=0
13050 WHILE a$<>""
13060 :
13070 :
13080 d=0:WHILE LEN(a$)<34 AND d=0
13090 tt$(x)=LEFT$(a$,LEN(a$)-1)
13100 a$=""
13110 d=1:WEND
13120 :
13130 :
13140 WHILE LEN(a$)>33
13150 FOR i=33 TO 1 STEP -1
13160 IF MID$(a$,i,1)<>" " THEN NEXT i:i
=33
13170 tt$(x)=LEFT$(a$,i-1)
13180 a$=MID$(a$,i+1)
13190 x=x+1
13200 WEND
13210 WEND
13220 x=x+1
13230 :
13240 :
13250 FOR i=ll+x TO pl+x STEP -1
13260 texte$(i)=texte$(i-x)
13270 NEXT i
13280 FOR i=0 TO x-1
13290 texte$(pl+i)=tt$(i)
13300 NEXT i
13310 a$=" ":p=0:ll=ll+x:pl=pl+x
```

Commentaires

Ligne 13030 : Cette ligne n'est là que pour faciliter la sortie du texte sur une imprimante. Le problème se pose lorsqu'il faut imprimer un ou plusieurs mots sur la partie droite de la page. L'affichage est basé sur des lignes de 32 caractères, tandis qu'une imprimante standard a des lignes de 80 caractères.

A l'impression, le programme Texte traite deux lignes du texte affiché ensemble pour créer une ligne imprimée. Donc si vous entrez du texte sur la partie droite de l'écran, vous obtiendrez soit une impression au milieu de la page soit un ajout à la fin de la ligne précédente. Pour pallier cet inconvénient, une méthode laborieuse consiste à entrer d'abord une ligne vide en appuyant sur ENTER, quand il n'y a pas de texte à entrer. Cela donne une ligne vierge sur le papier et garantit que la ligne imprimée suivante commencera du côté gauche de la page. Ensuite, il faut taper une ligne complète d'espaces, suivie d'une autre ligne d'espaces se terminant par le mot à placer du côté droit. L'imprimante 'imprimera' tous les espaces et la phrase finale apparaîtra sur la partie droite du document.

Toutefois, au programme final, la ligne complète d'espaces peut être remplacée par une ligne composée d'un simple "'. Le présent module traduit automatiquement cela en une ligne de 33 espaces.

Ligne 13040 : X est la variable qui permettra de détecter combien de lignes de texte sont nécessaires au nouveau texte ajouté.

Lignes 13050-13210 : Chaque fois qu'une ligne de caractères est ajoutée au bloc de texte principal, elle est découpée à partir de A$, le nouveau texte entré au module précédent. Ce processus se poursuit jusqu'à épuisement de A$.

Lignes 13080-13110 : Le nouveau texte est tout d'abord placé dans un tableau TT$ temporaire. S'il y a moins de 34 caractères y compris l'espace qui se trouve toujours à la fin, le nouveau texte peut tenir dans une ligne de 32 caractères. Il suffit de le transférer puis d'effacer A$.

Lignes 13140-13200 : S'il y a plus de 33 caractères, de sorte que le nouveau texte ne tient pas dans une seule ligne, il y a recherche vers l'arrière à partir du caractère 33 pour trouver le premier espace indiquant une fin de mot. La partie de A$ jusqu'à ce point peut alors être transférée ; cela empêche tout partage de mot en deux.

Cette section de texte disparaît du début de A$ et la boucle principale est à nouveau exécutée exactement de la même manière, le texte suivant étant placé dans une ligne différente de TT$.

Lignes 13250-13300 : La variable PL enregistre l'emplacement d'insertion de la nouvelle ligne dans le bloc principal de texte. Cet emplacement démarre à 0 quand il n'y a pas de texte, et reste normalement à la suite de ce qui a été entré. Des modu-

les ultérieurs vous permettront de déplacer le point d'insertion vers le haut et vers le bas dans le bloc principal. La première des deux boucles de cette section commence par libérer de la place au point d'insertion, pour le nombre de nouvelles lignes enregistrées par X. Pour cela, tout le texte est déplacé entre le point d'insertion et la dernière ligne (LL), vers le haut, du nombre de nouvelles lignes représenté par X. La deuxième boucle copie le contenu de TT$ dans l'espace ainsi créé.

Test

Entrez une ligne supplémentaire :

14130 RETURN

puis exécutez le programme. Lorsque le curseur clignotant apparaît, entrez :

AAA[ENTER]
BBB[ENTER]
CCC[ENTER]
DDD[ENTER]

Après cela, appuyez sur STOP pour mettre fin au programme. Puis, en mode direct, entrez :

FOR I=0 TO 5 :PRINT TEXTE$(I) :NEXT[ENTER]

Vous devez voir une ligne brisée, suivie des quatre lignes de texte entrées, suivies d'une autre ligne brisée indiquant la fin du texte.

Module 4.3.4 : Affichage du texte

Ce module affiche quinze lignes du bloc principal de texte.

MODULE 4.3.4 : LIGNES 14000-14130

```
14000 REM *************************
14010 REM *Affichage de la section*
14015 REM *       de texte        *
14020 REM *************************
14030 ss=pl-7
14040 IF ll-pl<8 THEN ss=ll-15
14050 IF ss<0 THEN ss=0
14060 CLS
14070 FOR i=ss TO ss+15
14080 PEN 2
14090 PRINT TEXTE$(I)
```

```
14100 IF i=pl-1 THEN PEN 1:PRINT ">"
14110 NEXT i
14120 LOCATE 1,23:PEN 2:PRINT a$:PEN 1
14130 RETURN
```

Commentaires

Lignes 14030-14050 : La variable SS représente la ligne de départ de la section à afficher (Section Start + Début de la section). Elle est calculée de façon à se trouver normalement huit lignes avant le point d'insertion actuel. Si le point d'insertion est près de la fin du texte, par exemple tout-à-fait en bas, cela signifie que huit lignes seulement seront affichées, donc la ligne 14040 fait reculer le point de départ. Enfin, s'il y a moins de quinze lignes ou si le point d'insertion est au début, le point de départ précédera le début du texte, donc il est augmenté jusqu'à 0.

Lignes 14070-14110 : Les 15 lignes sont à présent affichées. Toutes les lignes se terminent par un point-virgule pour empêcher que la position d'affichage ne descende d'une ligne. Sinon, une ligne pleine de 40 caractères (les première et dernière lignes comprenant des symboles graphiques) enverrait le curseur sur la ligne suivante, laissant une ligne vierge dans son déplacement. Pour des lignes comptant moins de 40 caractères, l'effet du point-virgule est annulé par un PRINT vide. Seul autre raffinement : le point d'insertion est marqué par un signe '>'; toute fonction d'édition telle qu'addition, copie, ou suppression de lignes agira sur la ligne au-dessous de la marque.

Ligne 14120 : Enfin, parallèlement à l'effacement de l'écran, tout contenu de A$ est réaffiché au bas de l'écran.

Test

Effectuez le même test qu'au dernier module, à la seule différence que, au lieu de devoir entrer une commande spéciale pour voir votre entrée, chaque ligne doit venir dans le texte principal lorsque vous appuyez sur ENTER.

Module 4.3.5 : Edition du texte principal

Nous avons vu comment éditer le nouveau texte pendant son entrée. Etudions à présent comment intervenir sur le texte principal existant. Ce module vous permet de déplacer le point

d'insertion sur le texte, de copier certaines lignes au bas de l'écran pour modification, et de supprimer des lignes existantes.

MODULE 4.3.5 : LIGNES 15000-15320

```
15000 REM **************
15010 REM *Mode Edition*
15020 REM **************
15030 WHILE 1
15040 p2=pl-ss
15050 :
15060 :
15070 t1$="":WHILE t1$=""
15080 LOCATE 1,p2+1:PRINT " ":FOR i=1 TO
 5:NEXT i
15090 LOCATE 1,p2+1:PRINT ">":FOR i=1 TO
 20:NEXT i
15100 t1$=LOWER$(INKEY$)
15110 WEND
15120 :
15130 :
15140 pl=pl+(t1$=CHR$(240))+10*(t1$="u")
:IF pl<1 THEN pl=1
15150 pl=pl-(t1$=CHR$(241))-10*(t1$="d")
:IF pl>ll THEN pl=ll
15160 IF t1$=CHR$(13) THEN t$="":RETURN
15170 :
15180 :
15190 WHILE pl<ll AND t1$=CHR$(127)
15200 FOR i=pl TO ll:texte$(i)=texte$(i+
1):NEXT i
15210 ll=ll-1
15220 t1$="":WEND
15230 :
15240 :
15250 IF pl<ll AND t1$="c" THEN a$=texte
$(pl)+" "
15260 IF t1$="p" OR t1$="v" THEN GOSUB 1
7000
15270 IF t1$="s" THEN GOSUB 18000
15280 IF t1$="f" THEN GOSUB 16000
15290 :
15300 :
15310 GOSUB 14000
15320 WEND
```

Commentaires

Ligne 15040 : Pour les besoins de ce module, le numéro de ligne sur l'écran du symbole '>', qui indique le point d'insertion en cours dans le texte, sera stocké dans la variable P2.

Lignes 15070-15110 : Le curseur '>' clignote pour indiquer que le programme est en mode édition.

Lignes 15140-15150 : Vous pouvez déplacer vers le haut ou vers le bas le curseur d'édition sur le texte, mais seulement en mode édition. La touche fléchée vers le haut ou vers le bas déplace le curseur d'une ligne. Les touches 'U' ou 'D' déplacent le curseur de 10 lignes vers le haut ou vers le bas. A noter que, en pratique, à moins que vous ne rencontriez le début ou la fin du fichier, ce n'est pas le curseur lui-même qui se déplace, mais le texte qui l'entoure.

Ligne 15160 : La touche ENTER met fin au mode édition.

Lignes 15190-15220 : La touche DEL actionnée en mode édition supprime la ligne de texte sous le curseur d'édition.

Ligne 15250 : La touche 'C' actionnée en mode édition copie la ligne se trouvant sous le curseur d'édition, au bas de l'écran, afin qu'elle puisse être traitée comme du nouveau texte.

Ligne 15260 : La touche 'P' ou 'V' actionnée en mode édition appelle le module suivant, qui envoie le texte à une imprimante ou à l'écran en mode 80 colonnes.

Ligne 15270 : La touche 'S' actionnée en mode édition appelle le module suivant qui sauvegarde le texte sur cassette.

Ligne 15280 : La touche 'F' actionnée en mode édition appelle le module suivant qui aménage le texte et essaie de standardiser autant que possible la longueur des lignes.

Test

Ce module étant entré, effectuez le même test que pour le module précédent, en insérant quatre groupes de lettres.

1) Puis frappez " et le curseur d'édition doit commencer à clignoter.
2) Entraînez-vous à déplacer le curseur d'édition vers le haut et vers le bas avec les touches fléchées. Si vous utilisez 'U' ou 'D', le curseur doit se porter au début ou à la fin du texte, puisqu'il y a moins de 10 lignes.
3) Déplacez le curseur d'édition sur l'avant-dernière ligne de texte. Frappez 'C' ; 'DDD' doit être copié sur la partie inférieure de l'écran.

4) Appuyez sur la touche DEL et la ligne 'DDD' doit disparaître du texte principal.
5) Déplacez le curseur d'édition au début du texte.
6) Appuyez sur ENTER pour terminer le mode édition.
7) Lorsque le curseur clignotant revient au bas de l'écran, appuyez à nouveau sur ENTER. 'DDD' doit se trouver au début du texte.

Module 4.3.6 : Formatage du texte

Vous pouvez également aménager le bloc principal de texte entré. C'est nécessaire car l'un des principaux avantages de l'édition de texte, ou de son aîné, le traitement de texte, est la possibilité de supprimer ou d'insérer de nouvelles phrases dans un texte existant. Après quoi, le texte paraît souvent désordonné; c'est pourquoi la plupart des logiciels de traitement de texte qui n'aménagent pas automatiquement le texte pendant l'entrée, disposent d'une fonction 'format'. Vue au niveau le plus élémentaire, la fonction de formatage consiste à examiner chaque ligne et à déterminer si le premier mot de la ligne suivante pourrait être placé à la fin de celle-ci; si c'est le cas, le mot est déplacé. Il en résulte un texte plus net, même si des espaces ne sont pas insérés pour que toutes les lignes se terminent dans le même alignement. Le présent module assure le formatage du texte contenu dans TEXTE$, en utilisant les techniques de découpage de chaîne présentées au chapitre 1.

MODULE 4.3.6 : LIGNES 16000-16310

```
16000 REM **********************
16010 REM *Formatage de lignes*
16020 REM **********************
16030 FOR i=1 TO ll-2
16040 d=0:WHILE texte$(i)<>"" AND texte$
(i+1)<>"" AND texte$(i)<>SPACE$(32) AND
texte$(i+1)<>SPACE$(32) AND d=0
16050 :
16060 :
16070 sp=32-LEN(texte$(i))
16080 mot=INSTR(texte$(i+1)," ")
16090 IF mot=0 THEN mot=LEN(texte$(i+1))
+1
16100 IF mot>sp THEN d=1
16110 :
16120 :
16130 d2=0:WHILE sp>=mot AND sp<=LEN(tex
te$(i+1)) AND d2=0
```

```
16140 texte$(i)=texte$(i)+" "+LEFT$(text
e$(i+1),mot-1)
16150 texte$(i+1)=MID$(texte$(i+1),mot+1
)
16160 d2=1:WEND
16170 :
16180 :
16190 sp=32-LEN(texte$(i))
16200 d2=0:WHILE LEN(texte$(i+1))<sp AND
 d=0 AND d2=0
16210 texte$(i)=texte$(i)+" "+texte$(i+1
)
16220 FOR j=i+1 TO ll
16230 texte$(j)=texte$(j+1)
16240 NEXT j
16250 ll=ll-1:pl=pl-1
16260 d2=1:WEND
16270 :
16280 :
16290 WEND
16300 NEXT i
16310 RETURN
```

Commentaires

Lignes 16030-16300 : L'opération de formatage commence à la première ligne et se poursuit ligne à ligne sur tout le texte.

Lignes 16040-16290 : Seules sont concernées les lignes ayant un contenu. Par exemple, à la fin d'un paragraphe, il peut y avoir une ligne inférieure à 32 caractères ; nous ne souhaitons pas voir le début du paragraphe suivant s'ajouter à la fin de cette ligne. Pour cela, il suffit d'entrer une ligne vide après la fin du paragraphe. Le module de formatage n'ajoute pas cette ligne vide à la fin de la ligne précédente, et il ne copie pas non plus des mots de la ligne suivante, dans la ligne vide.

Ligne 16070 : La variable SP permet de stocker la quantité d'espace restant à la fin de la ligne en cours d'examen.

Ligne 16080 : MOT permet de stocker la longueur du premier mot sur la ligne suivante, comme l'indique la présence d'un espace.

Ligne 16090 : S'il n'y a pas d'espace sur la ligne suivante, la longueur de MOT devient identique à la longueur de la ligne complète.

Ligne 16100 : Si la longueur du premier mot sur la ligne suivante est supérieure à l'espace disponible à la fin de la ligne en cours, il est inutile de poursuivre cette dernière plus avant. Donc la boucle WHILE se termine et la boucle FOR passe à la ligne de texte suivante.

Lignes 16130-16160 : Ces lignes sont mises en action s'il y a assez de place à la fin de la ligne en cours pour le mot suivant, mais pas pour toute la ligne suivante. Elles ont pour effet de copier le mot à la fin de la ligne en cours et de le séparer du début de la ligne suivante.

Lignes 16190-16260 : Peut-être la ligne en cours a-t-elle assez de place pour contenir la totalité de la ligne suivante. Dans ce cas, il ne suffit pas de copier la ligne suivante dans la ligne en cours : la totalité du tableau doit être décalée d'une ligne pour combler l'espace créé, sinon le programme sera gêné par l'instruction, citée précédemment, que les lignes vides doivent rester indépendantes.

Test

Effectuez le même test que précédemment, en entrant les quatre groupes de lettres. Lorsqu'elles sont toutes affichées comme faisant partie du texte principal, frappez " pour entrer en mode édition puis frappez 'F'. Après un certain délai (qui sera relativement long si vous avez entré beaucoup de texte), le texte principal s'affiche à nouveau, mais cette fois-ci les quatre groupes de lettres ont été rassemblés sur une ligne.

Module 4.3.7 : Fichiers de données

Il s'agit d'un module de fichier de données standard. A noter l'utilisation de LINE INPUT, qui assure que le texte contenant des virgules et d'autres caractères sur lesquels bute INPUT, est bien pris en compte.

En outre, ce module semble avoir mis en évidence une anomalie assez mystérieuse sur la plupart des 464. Elle se traduit par une altération, totalement invisible pour l'utilisateur, de la chaîne FI$ pendant le déroulement du module ; il s'ensuit que le 464 ne reconnaît pas le nom du fichier sur cassette comme étant le même que celui qui a été entré, même si les deux s'affichent de façon identique. Dans notre propre version du programme, la ligne suivante :

```
19075FI$=FI$
```

aussi curieux que cela puisse paraître, contourne le problème en rappelant à l'ordinateur la vraie valeur de la chaîne. Si le programme fonctionne sans cet ajout (comme il le devrait), vous pouvez ignorer ce commentaire.

MODULE 4.3.7 : LIGNES 18000-19150

```
18000 REM *************************
18010 REM *Sauvegarde sur cassette*
18020 REM *************************
18030 q$="":WHILE LOWER$(q$)<>"o"
18040 CLS:INPUT "Nom de fichier:",fi$
18050 PRINT "Fichier à sauvegarder ";fi$
18060 INPUT "Est-ce correct (o/n)";q$
18070 WEND
18080 OPENOUT fi$
18090 PRINT #9,pl
18100 PRINT #9,ll
18110 FOR i=0 TO ll
18130 PRINT #9,texte$(i)
18140 NEXT i
18150 CLOSEOUT
18160 RETURN
19000 REM **********************
19010 REM *Lecture sur cassette*
19020 REM **********************
19030 q$="":WHILE LOWER$(q$)<>"o"
19040 PRINT:INPUT "Nom de fichier:",fi$
19050 PRINT "Fichier à sauvegarder ";fi$
19060 INPUT "Est-ce correct (o/n)";q$
19070 WEND
19080 OPENIN fi$
19090 INPUT #9,pl
19095 INPUT #9,ll
19100 FOR i=0 TO ll
19110 LINE INPUT #9,texte$(i)
19130 NEXT i
19140 CLOSEIN
19150 RETURN
```

Module 4.3.8 : Module de contrôle

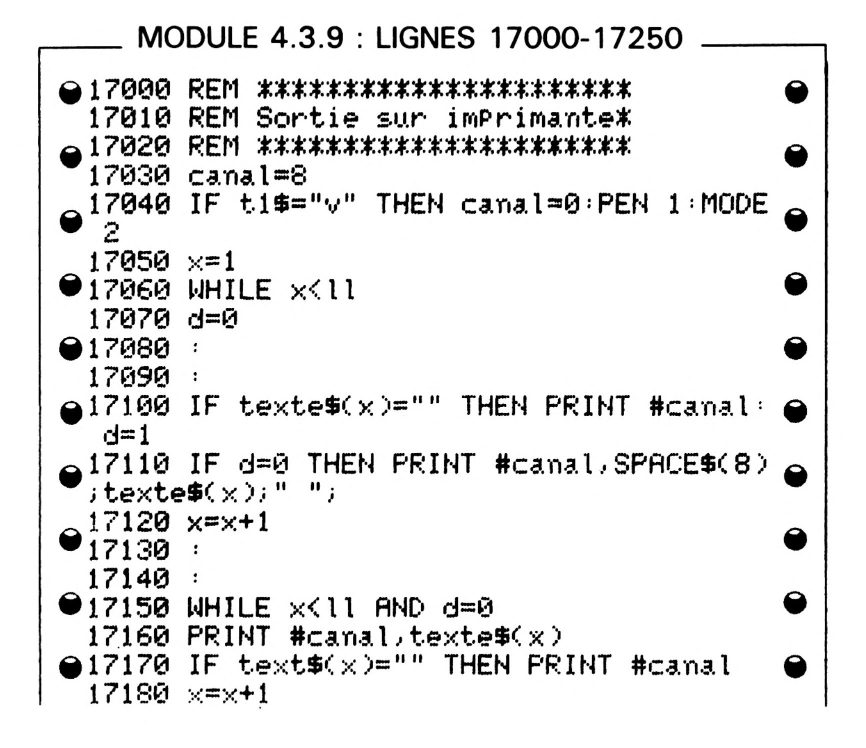
Il s'agit d'un module de contrôle standard.

MODULE 4.3.8 : LIGNES 10000-10060

```
10000 REM *******************
10010 REM *Boucle de Controle*
10020 REM *******************
10030 MODE 1
10040 IF in=0 THEN GOSUB 11000
10050 GOSUB 14000
10060 GOTO 12000
```

Module 4.3.9 : Impression du texte

Si vous disposez d'une imprimante - et un éditeur de texte vous sera bien peu utile si vous n'en avez pas - il vous permet de transférer le contenu de l'écran sur papier. En outre, ce module vous permet de visualiser d'abord le texte en mode 80 colonnes sur l'écran, pour vous assurer que sa présentation donnera un bon résultat sur l'imprimante.



MODULE 4.3.9 : LIGNES 17000-17250

```
17000 REM **********************
17010 REM Sortie sur imprimante*
17020 REM **********************
17030 canal=8
17040 IF t1$="v" THEN canal=0:PEN 1:MODE
 2
17050 x=1
17060 WHILE x<11
17070 d=0
17080 :
17090 :
17100 IF texte$(x)="" THEN PRINT #canal:
 d=1
17110 IF d=0 THEN PRINT #canal,SPACE$(8)
;texte$(x);" ";
17120 x=x+1
17130 :
17140 :
17150 WHILE x<11 AND d=0
17160 PRINT #canal,texte$(x)
17170 IF text$(x)="" THEN PRINT #canal
17180 x=x+1
```

```
17185 IF canal=0 THEN IF INKEY$="" THEN
GOTO 17185
17190 d=1:WEND
17200 :
17210 :
17220 WEND
17230 PRINT #canal
17240 IF canal=0 THEN IF INKEY$="" THEN
GOTO 17240
17250 MODE 1 : RETURN
```

Commentaires

Lignes 17030-17040 : Ce module est appelé, que vous souhaitiez voir le texte sur une imprimante ou sur l'écran en mode 80 colonnes. Toutefois, si vous voulez l'afficher sur l'écran, la sortie du programme sera dirigée vers le canal 0 (l'écran) et le MODE2 sera actif.

Ligne 17050 : La variable X permet d'enregistrer la progression du module au travers des lignes de TEXTE$.

Ligne 17070 : La variable D permettra de sauter certains segments du module lorsqu'il ne reste plus rien à faire sur une ligne particulière de texte.

Ligne 17100 : Comme lors du formatage, le programme ne tente pas de traiter des lignes vides; lorsqu'il en rencontre une, une instruction PRINT # vide permet de déplacer la position d'impression d'une ligne vers le bas. La variable D reçoit la valeur 1 pour indiquer qu'une ligne de texte est terminée.

Ligne 17110 : Si la ligne en cours n'est pas vide, elle s'imprime dans la première moitié d'une ligne de 64 caractères, suivie d'un espace. Notez le point-virgule qui assure que les caractères suivants imprimés suivront immédiatement.

Lignes 17150-17190 : La deuxième moitié de la ligne est imprimée. S'il s'agissait d'une ligne vide, un PRINT # supplémentaire permet de déplacer l'imprimante sur la ligne suivante. Si la sortie est envoyée à l'écran, un temps d'arrêt est inséré après chaque ligne affichée, jusqu'à ce que vous frappiez une touche. Cela permet d'afficher des documents dont la longueur dépasse une page d'écran.

Ligne 17240 : Là encore, si l'affichage a lieu sur l'écran, une autre frappe est nécessaire lorsque le document est terminé. Cela vous empêche de dépasser la dernière ligne par mégarde et de perdre ainsi l'affichage du document.

Test

i vous disposez d'une imprimante, vérifiez qu'elle est bien connectée et sous tension, puis entrez du texte. Appuyez sur SHIFT/0 pour entrer en mode édition puis frappez 'P'. Le texte doit s'imprimer. Si ce test est satisfaisant le programme est opérationnel. Pour vous aider à bien maîtriser le programme, voici un tableau des différentes commandes d'une touche.

TEXTE : TABLEAU DES COMMANDES D'UNE TOUCHE

En mode entrée de texte

ENTER	Place le nouveau texte dans le texte principal.
DEL	Déplace le curseur au début ou à la fin de la ligne.
TOUCHES FLECHEES	Supprime le caractère à gauche du curseur.
^	Déplacent le curseur à gauche ou à droite. Pour entrer en mode édition.

En mode édition

ENTER	Met fin au mode édition.
TOUCHES FLECHEES	Déplacent le curseur d'édition d'une ligne vers le haut ou vers le bas.
U ou D	Déplacent le curseur d'édition de 10 lignes vers le haut ou vers le bas.
DEL	Supprime la ligne située directement sous le curseur d'édition.
C	Copie la ligne située directement sous le curseur d'édition.
P	Imprime le texte principal en cours.
S	Sauvegarde le texte principal en cours sur cassette.
F	Formate le texte principal.
V	Visualise le texte en format 80 colonnes

PROGRAMME 4.4 : MULTIQ

Fonction du programme

Le dernier programme de ce chapitre va nous permettre de nous éduquer en joignant l'utile à l'agréable. Je ne sais pas si vous apprendrez beaucoup sur un sujet donné avec ce programme, mais il est distrayant à utiliser et vous donnera le virus des 'questions/réponses'. De plus, ce programme a ses propres lettres de noblesse car une version précédente de MultiQ a été le premier programme (tout au moins d'après le communiqué de presse) à la base d'un jeu de questions pour les auditeurs à la radio britannique.

Comme son nom le laisse deviner, MultiQ est un programme polyvalent. Il peut vous apprendre une langue et, quelques minutes plus tard, vous poser des questions très difficiles sur l'histoire du 19ème siècle. Il crée des tests avec questions/réponses aléatoires du type de plus en plus répandu dans les examens, posant une question et fournissant cinq réponses possibles, dont une seule est correcte. Un score permanent est tenu à jour qui permet d'évaluer les connaissances de l'utilisateur sur le sujet en question. Bien entendu, l'essentiel du travail incombe au programmeur ; non seulement il faut entrer le programme, mais également donner un groupe de questions suffisamment vaste pour que les tests soient significatifs.

Module 4.4.1 : Initialisation

Comme pour tous les programmes polyvalents de ce chapitre, ce module permet de décrire le type de fichier à gérer dans la session en cours.

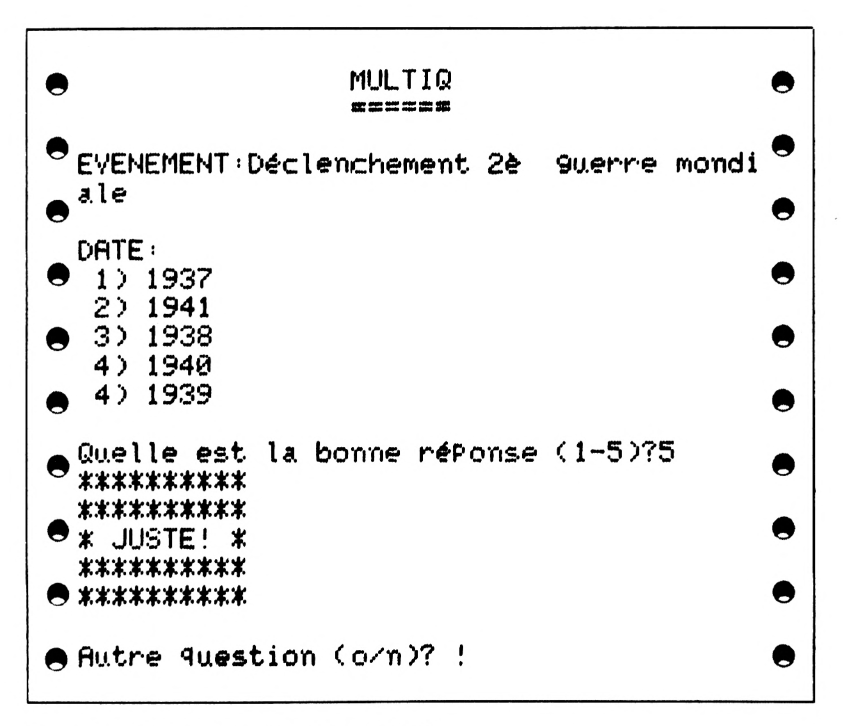
MODULE 4.4.1 : LIGNES 10000-10100

```
10000 REM ****************
10010 REM *Initialisation*
10020 REM ****************
10030 DIM nom$(1),qu(4),tableau$(499,1),
qtypes$(9),ntype(1,9)
10040 juste=0:total=0:it=0
10050 MODE 1
10060 PEN 2:PRINT TAB(17);"MULTIQ":PRINT
 TAB(17);"======"
```

```
10070 WINDOW 1,40,4,25
10080 PEN 1
10090 INPUT "Lire sur cassette ? (o/n)";
q$
10100 IF LOWER$(q$)="o" THEN GOSUB 23000
 ELSE GOSUB 24000
```

Commentaires

Ligne 10030 : Le tableau NOM$ permet d'enregistrer les noms généraux donnés par l'utilisateur aux questions et aux réponses, QU sera utilisé pour définir



```
MULTIQ
======

EVENEMENT:Déclenchement 2è  guerre mondi
ale

DATE:
 1) 1937
 2) 1941
 3) 1938
 4) 1940
 4) 1939

Quelle est la bonne réponse (1-5)?5
**********
**********
* JUSTE! *
**********
**********

Autre question (o/n)? !
```

Figure 4.4 : Partie d'un test créé par MultiQ

les tests aléatoires, TABLEAU$ contiendra le fichier principal de questions et de réponses, QTYPES contiendra les noms des types pouvant être alloués aux questions et aux réponses, et NTYPE stockera le nombre de chaque type dans le fichier principal.

Module 4.4.2 : Structure du test

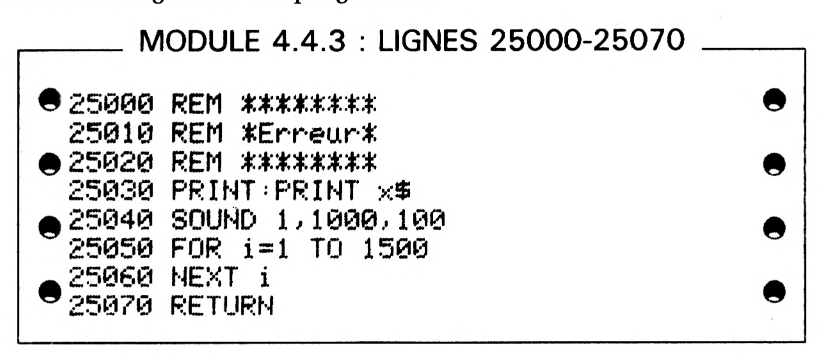
Nous avons dit que MultiQ définit des tests, c'est-à-dire pose des questions et affiche les réponses possibles. Ces lignes vous permettent de donner un titre général aux questions et aux réponses. Par exemple, si le programme devait servir à apprendre l'anglais, vous pourriez poser la question 'MOT FRANCAIS' et la réponse 'MOT ANGLAIS EQUIVALENT'.

MODULE 4.4.2 : LIGNES 24000-24130

```
24000 REM *******************
24010 REM *Structure du Test*
24020 REM *******************
24030 q$="":WHILE LOWER$(q$)<>"o"
24040 CLS
24050 PRINT "Structure du Test:":PRINT
24060 INPUT "Nom de la réponse:",nom$(0)
24070 INPUT "Nom de la question:",nom$(1
)
24080 PRINT:INPUT "Est-ce correct ? (o/n
)";q$
24090 WEND
24100 qtype$(0)="Pas de type"
24110 ntypes=1
24120 GOSUB 12000
24130 RETURN
```

Module 4.4.3 : Messages d'erreur

Ce module standard affiche les messages d'erreur provenant d'autres segments du programme.



MODULE 4.4.3 : LIGNES 25000-25070

```
25000 REM ********
25010 REM *Erreur*
25020 REM ********
25030 PRINT:PRINT x$
25040 SOUND 1,1000,100
25050 FOR i=1 TO 1500
25060 NEXT i
25070 RETURN
```

Module 4.4.4 : Le menu

Il s'agit d'un module de menu standard

MODULE 4.4.4. : LIGNES 11000-11210

```
11000 REM ***************
11010 REM *Menu Principal*
11020 REM ***************
11030 WHILE z<>7
11040 CLS
11050 PRINT"COMMANDES DISPONIBLES:"
11060 PRINT
11070 PRINT "1) Entrer nouvelles données
"
11080 PRINT "2) Entrer nouveaux types"
11090 PRINT "3) Rechercher/Supprimer"
11100 PRINT "4) Générer les questions"
11110 PRINT "5) Afficher ou réinitialise
r le résultat"
11120 PRINT "6) Sauvegarder les données
sur cassette"
11130 PRINT "7) Arreter"
11140 PRINT
11150 INPUT"Votre choix: ",z
11160 IF z>2 AND z<7 AND it=0 THEN x$="*
******** PAS ENCORE DE DONNEES ********
":GOSUB 25000:z=0
11170 ON z GOSUB 13000,12000,20000,17000
,19000,22000
11180 WEND
11190 CLS
11200 LOCATE 12,11:PRINT "FIN DU COURS"
11210 END
```

Module 4.4.5 : Définition des types de questions

Lorsque vous entrerez les modules ultérieurs, vous constaterez que MultiQ prévoit deux niveaux de difficulté dans ses tests. Il utilise pour cela des types de questions. Si nous reprenons l'exemple d'un cours d'anglais, il est possible de diviser les types de mots affichés en différents groupes grammaticaux, tels que verbes, noms, adjectifs, et ainsi de suite. Si un mot fran-

çais affiché est un verbe, et si, parmi les cinq réponses anglaises possibles une seule est un verbe, le test est bien plus facile que si les cinq réponses possibles étaient des verbes.

Le présent module vous permet de définir jusqu'à 10 types de questions ou de réponses, avec la possibilité d'associer un type à une question dès son entrée. Lorsque MultiQ définit ultérieurement un test, il vous demande si vous souhaitez que les réponses possibles ne proviennent que du même type que la réponse correcte, ou de l'ensemble des réponses.

MODULE 4.4.5 : LIGNES 12000-12150

```
12000 REM ***************
12010 REM *Nouveaux types*
12020 REM ***************
12030 q$="":WHILE 1
12040 CLS
12050 PRINT "Types:":PRINT
12060 PRINT "Types jusque là:":PRINT
12070 FOR i=0 TO ntypes-1
12080 PRINT i+1;") ";qtype$(i)
12090 NEXT i
12100 IF ntypes=10 THEN x$="**** PLUS DE
 PLACE POUR D'AUTRES TYPES ****":GOSUB 2
5000:RETURN
12110 PRINT:INPUT "Entrer nouveau type (
'ç' pour quitter):";q$
12120 IF q$="ç" THEN RETURN
12130 qtype$(ntypes)=q$
12140 ntypes=ntypes+1
12150 WEND
```

Test

Vous devez être en mesure d'exécuter le programme et d'entrer jusqu'à 10 types de questions. Pour avoir confirmation que les types ont bien été acceptés, arrêtez le programme et tapez :

FOR I=0 TO 9 :PRINT QTYPE$(I) :NEXT I[ENTER]

Module 4.4.6 : Recherche binaire

Il s'agit d'un module de recherche standard qui fonctionne d'après l'ordre alphabétique selon des réponses à des questions. A noter que, comme vous le constaterez en entrant le module des nouvelles données dans un moment, chaque

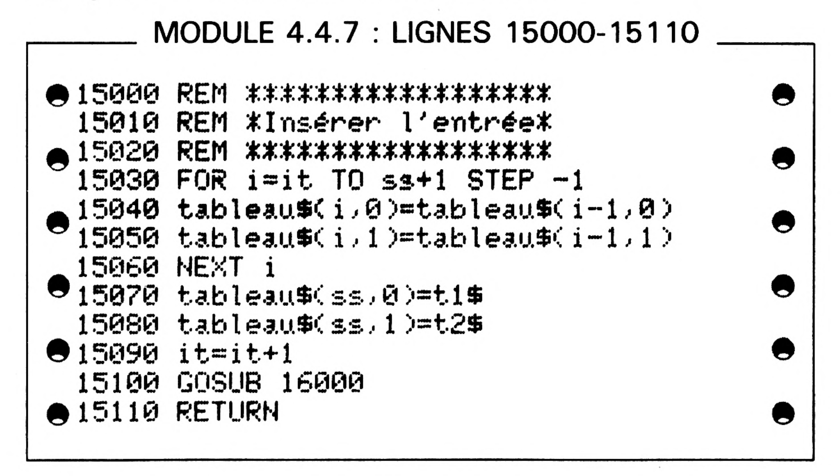
réponse est précédée d'un simple caractère (0-9) pour indiquer son type. Le fichier sera donc trié d'abord d'après les types et à l'intérieur des types, d'après l'ordre alphabétique.

MODULE 4.4.6 : LIGNES 14000-14110

```
14000 REM ******************
14010 REM *Recherche binaire*
14020 REM ******************
14030 IF it=0 THEN ss=0:RETURN
14040 po=INT(LOG(it)/LOG(2)):ss=2^po-1
14050 FOR i=po TO 0 STEP -1
14060 ss=ss+2^i*((tableau$(ss,0)>t1$)-(t
ableau$(ss,0)<t1$))
14070 IF ss<0 THEN ss=0
14080 IF ss>it-1 THEN ss=it-1
14090 NEXT i
14100 IF tableau$(ss,0)<t1$ THEN ss=ss+1
14110 RETURN
```

Module 4.4.7 : Insertion d'une donnée

Il s'agit d'un module d'insertion standard.



MODULE 4.4.7 : LIGNES 15000-15110

```
15000 REM *****************
15010 REM *Insérer l'entrée*
15020 REM *****************
15030 FOR i=it TO ss+1 STEP -1
15040 tableau$(i,0)=tableau$(i-1,0)
15050 tableau$(i,1)=tableau$(i-1,1)
15060 NEXT i
15070 tableau$(ss,0)=t1$
15080 tableau$(ss,1)=t2$
15090 it=it+1
15100 GOSUB 16000
15110 RETURN
```

Module 4.4.8 : Suivi des types

Nous avons entré le module permettant d'enregistrer des types. Mais il faut aussi que le programme puisse connaître le nombre de chaque type se trouvant dans le fichier, et où commence chaque groupe de types.

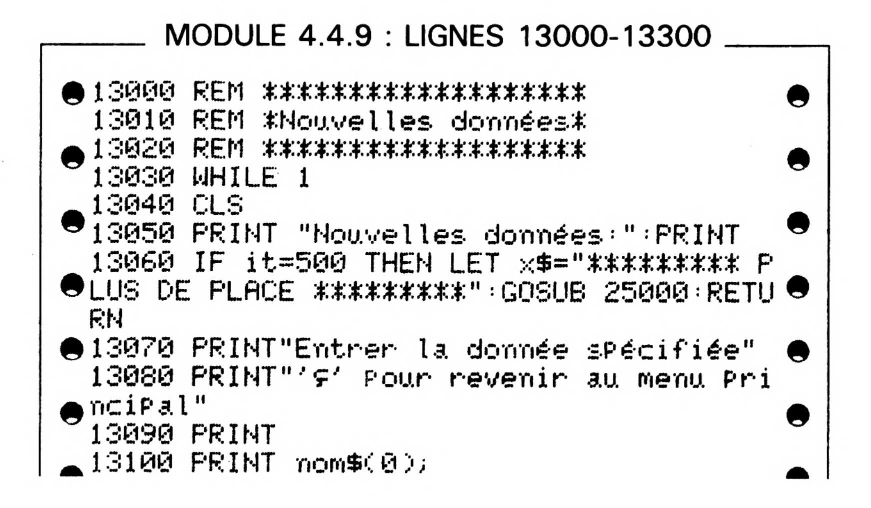
L'enregistrement du nombre de chacun des types est effectué par le module d'entrée qui ajoute simplement 1 à l'élément concerné dans le tableau NTYPE. Ce module est appelé pour chaque nouvelle entrée ou suppression. Il a pour but d'enregistrer de l'autre côté de NTYPE les totaux cumulés des types de 0 à 9. Le cas échéant, les réponses seront classées dans l'ordre des types à l'intérieur du fichier principal ; ainsi, si nous connaissons le total des données appartenant aux types de 0 à 2, nous savons où commencent les données du groupe 3.

MODULE 4.4.8 : LIGNES 16000-16080

```
16000 REM ************
16010 REM *Mise à jour*
16020 REM ************
16030 su=0
16040 FOR i=0 TO 9
16050 ntype(1,i)=su
16060 su=su+ntype(0,i)
16070 NEXT i
16080 RETURN
```

Module 4.4.9 : Entrée d'un nouvel article

Il s'agit d'un module simple qui vous permet d'entrer un nouveau couple question/réponse, et de lui associer un type.



MODULE 4.4.9 : LIGNES 13000-13300

```
13000 REM *******************
13010 REM *Nouvelles données*
13020 REM *******************
13030 WHILE 1
13040 CLS
13050 PRINT "Nouvelles données:":PRINT
13060 IF it=500 THEN LET x$="********* P
LUS DE PLACE *********":GOSUB 25000:RETU
RN
13070 PRINT"Entrer la donnée spécifiée"
13080 PRINT"'ç' pour revenir au menu pri
ncipal"
13090 PRINT
13100 PRINT nom$(0);
```

```
13110 INPUT ":";t1$
13120 IF t1$="ç" THEN RETURN
13130 PRINT nom$(1);
13140 INPUT ":";t2$
13150 IF t2$="ç" THEN RETURN
13160 PRINT:PRINT "Types:":PRINT
13170 FOR i=0 TO ntypes-1
13180 PRINT i+1;") ";qtype$(i)
13190 NEXT i
13200 PRINT:INPUT "Entrer le numéro du t
ype:",t3
13210 t3=t3-1
13220 IF t3<0 OR t3>ntypes THEN t3=0
13230 PRINT:INPUT "Est-ce correct (o/n)"
;q$
13240 WHILE LOWER$(q$)="o"
13250 ntype(0,t3)=ntype(0,t3)+1
13260 t1$=MID$(STR$(t3)+t1$,2)
13270 GOSUB 14000
13280 GOSUB 15000
13290 q$="":WEND
13300 WEND
```

Commentaires

Ligne 13250 : L'élément du tableau NTYPE représentant le type spécifié pour la question/réponse en cours est augmenté de 1. C'est ce tableau, nous l'avons vu, qui est élaboré par le module de mise à jour lorsqu'il enregistre où commence chaque groupe dans le tableau.

Test

A ce stade, vous pouvez exécuter le programme, spécifier les types puis sélectionner l'option 1 du menu pour commencer à entrer des questions et des réponses. Pour vérifier que les données sont correctement reçues, entrez ce qui suit :

QUESTION	REPONSE	TYPE
Q111	A111	3
Q222	A222	2
Q333	A333	1

puis sortez du programme. Ensuite tapez :

FOR I=0 TO 2 :FOR J=0 TO 1 :PRINT TABLEAU$(I,J) :NEXT J :NEXT I[ENTER]

Vous devez obtenir :

0A333
Q333
1A222
Q222
2A333
Q111

Module 4.4.10 : Stockage des données

A présent que nous pouvons entrer des données, nous allons passer aux deux modules standard permettant de stocker ces données et de les lire.

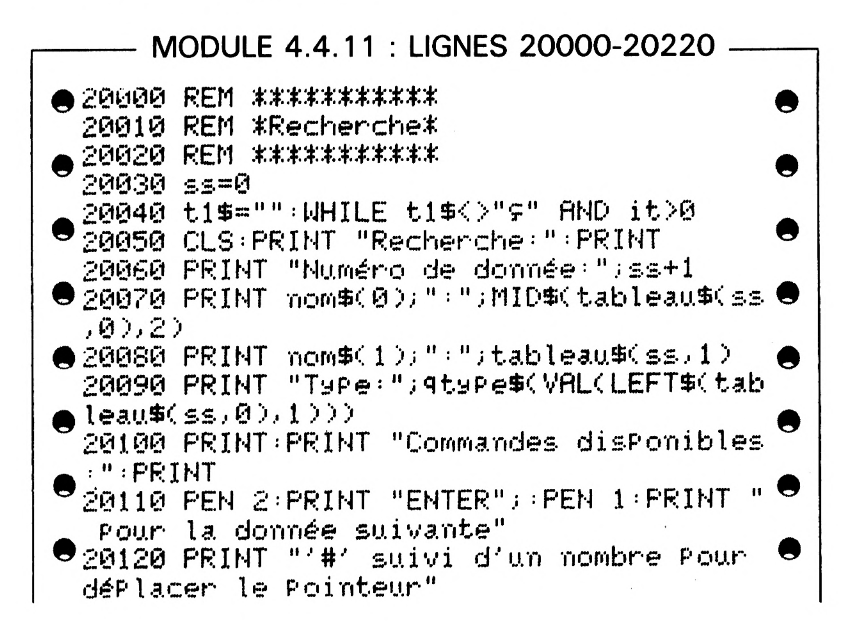
MODULE 4.4.10 : LIGNES 22000-23220

```
22000 REM *************************
22010 REM *Sauvegarde sur cassette*
22020 REM *************************
22030 CLS
22040 PRINT "Sauvegarder les données:":P
RINT
22050 INPUT "Nom du fichier:",fichier$
22060 OPENOUT fichier$
22070 PRINT #9,it
22080 PRINT #9,ntypes
22090 FOR i=0 TO 1
22100 PRINT #9,nom$(i)
22110 FOR j=0 TO ntypes-1
22120 PRINT #9,ntype(i,j)
22130 NEXT j
22140 FOR j=0 TO it-1
22150 PRINT #9,tableau$(j,i)
22160 NEXT j
22170 NEXT i
22180 FOR i=0 TO ntypes-1
22190 PRINT #9,qtype$(i)
22200 NEXT i
22210 CLOSEOUT
22220 RETURN
23000 REM **********************
23010 REM *Lecture sur cassette*
23020 REM **********************
23030 CLS
23040 PRINT "charger les données":PRINT
23050 INPUT "Nom du fichier:",fichier$
```

```
23060 OPENIN fichier$
23070 INPUT #9,it
23080 INPUT #9,ntypes
23090 FOR i=0 TO 1
23100 INPUT #9,nom$(i)
23110 FOR j=0 TO ntypes-1
23120 INPUT #9,ntype(i,j)
23130 NEXT j
23140 FOR j=0 TO it-1
23150 INPUT #9,tableau$(j,i)
23160 NEXT j
23170 NEXT i
23180 FOR i=0 TO ntypes-1
23190 INPUT #9,qtype$(i)
23200 NEXT i
23210 CLOSEIN
23220 RETURN
```

Module 4.4.11 : Recherche par l'utilisateur

Il s'agit d'un simple module de recherche qui vous permet de faire une recherche vers l'avant et vers l'arrière dans le fichier principal, d'afficher des données et, après l'entrée du module suivant, de les supprimer.



MODULE 4.4.11 : LIGNES 20000-20220

```
20000 REM ***********
20010 REM *Recherche*
20020 REM ***********
20030 ss=0
20040 t1$="":WHILE t1$<>"ç" AND it>0
20050 CLS:PRINT "Recherche:":PRINT
20060 PRINT "Numéro de donnée:";ss+1
20070 PRINT nom$(0);":";MID$(tableau$(ss
,0),2)
20080 PRINT nom$(1);":";tableau$(ss,1)
20090 PRINT "Type:";qtype$(VAL(LEFT$(tab
leau$(ss,0),1)))
20100 PRINT:PRINT "Commandes disponibles
:":PRINT
20110 PEN 2:PRINT "ENTER";:PEN 1:PRINT "
 pour la donnée suivante"
20120 PRINT "'#' suivi d'un nombre pour
déplacer le pointeur"
```

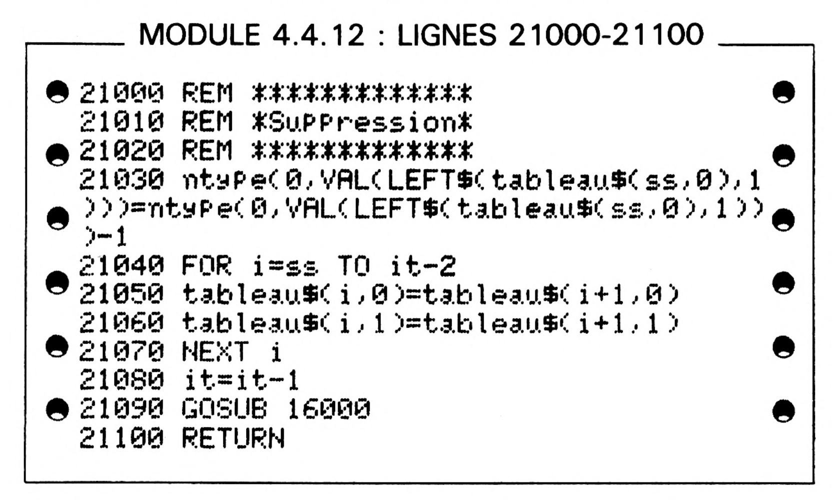
```
20130 PRINT "'d' pour effacer la donnée"
20140 PRINT "'q' pour quitter"
20150 PRINT:INPUT "Que voulez-vous";t1$
20160 IF t1$="" THEN t1$="#1"
20170 IF LOWER$(t1$)="d" THEN GOSUB 2100
0
20180 IF LEFT$(t1$,1)="#" THEN ss=ss+VAL
(MID$(t1$,2))
20190 IF ss>it-1 THEN ss=it-1
20200 IF ss<0 THEN ss=0
20210 WEND
20220 RETURN
```

Test

Exécutez le programme et lisez les données stockées sur cassette. Avec l'option 3 du menu, vous pouvez consulter les données vers l'avant et vers l'arrière.

Module 4.4.12 : Suppression d'une donnée

Il s'agit d'un module de suppression standard.



MODULE 4.4.12 : LIGNES 21000-21100

```
21000 REM ************
21010 REM *Suppression*
21020 REM ************
21030 ntype(0,VAL(LEFT$(tableau$(ss,0),1
)))=ntype(0,VAL(LEFT$(tableau$(ss,0),1))
)-1
21040 FOR i=ss TO it-2
21050 tableau$(i,0)=tableau$(i+1,0)
21060 tableau$(i,1)=tableau$(i+1,1)
21070 NEXT i
21080 it=it-1
21090 GOSUB 16000
21100 RETURN
```

Test

Suivez la même procédure qu'au module précédent, mais frappez 'S' en face de l'une des données. Vous constaterez qu'elle a été supprimée.

Module 4.4.13 : Définition des questions

Nous en arrivons au module qui constitue l'originalité de MultiQ, en définissant les tests à choix multiples. Ce module traite la partie visible de l'opération : l'affichage des questions et des réponses possibles et le choix d'une réponse correcte par l'utilisateur.

MODULE 4.4.13 : LIGNES 17000-17430

```
17000 REM ***********
17010 REM *Questions*
17020 REM ***********
17030 CLS
17040 PRINT "Questions:":PRINT
17050 PRINT "           Voulez-vous que le
s réponses ne proviennent que";
17060 PRINT "d'un type (difficile) ou de
  l'ensemble"
17070 PRINT "stock (facile)?"
17080 PRINT:PRINT "1) Un type seulement"
17090 PRINT "2) Tous les types"
17100 PRINT:INPUT "Lequel:",rq
17110 IF rq<1 OR rq>2 THEN rq=2
17120 q$="":WHILE LOWER$(q$)<>"n"
17130 GOSUB 18000
17140 CLS
17150 PRINT nom$(1);":";tableau$(qu(qPos
),1)
17160 PRINT:PRINT nom$(0);":"
17170 FOR i=0 TO 4
17180 PRINT i+1;") ";MID$(tableau$(qu(i)
,0),2)
17190 NEXT i
17200 ra=0:WHILE ra<1 OR ra>5
17210 PRINT:INPUT "Quelle est la bonne r
éponse (1-5)";ra
17220 WEND
17230 WHILE ra-1=qPos
17240 PEN 3:PRINT
17250 PRINT "**********"
17260 PRINT "*        *"
17270 PRINT "* JUSTE! *"
17280 PRINT "*        *"
17290 PRINT "**********"
17300 FOR i=800 TO 100 STEP -100
17310 SOUND 1,i,10
```

```
17320 NEXT i
17330 PEN 1
17340 juste=juste+1
17350 ra=0:WEND
17360 WHILE ra-1<>qPos AND ra>0
17370 PRINT:PRINT "Désolé, c'est faux. L
a bonne réponse"
17380 PRINT "était ";MID$(tableau$(mainq
,0),2);"."
17390 ra=0:WEND
17400 total=total+1
17410 PRINT:INPUT "Autre question (o/n)"
;q$
17420 WEND
17430 RETURN
```

Commentaires

Lignes 17040-17100 : Nous avons déjà vu que MultiQ permet deux niveaux de tests. Ces lignes vous permettent de spécifier si les réponses possibles doivent être extraites du fichier tout entier ou d'un type de réponse particulier.

Lignes 17150-17190 : La question et les cinq réponses possibles, qui seront sélectionnées par le module suivant, sont affichées. Les positions des cinq réponses possibles sont contenues dans le tableau QU et la position de la réponse correcte dans QU est enregistrée par la variable QPOS.

Lignes 17230-17350 : Si la réponse choisie, telle qu'elle est représentée par la variable RA, correspond à la position de la réponse correcte (QPOS), le mot 'JUSTE' est affiché et un bref signal sonore retentit. La variable JUSTE qui enregistre le nombre de réponses justes est augmentée de 1. En cas de mauvaise réponse, vous en êtes informé et la bonne réponse vous est indiquée.

Ligne 17400 : TOTAL, la variable qui enregistre le nombre total de questions posées, est augmentée de 1.

Module 4.4.14 : Sélection des questions aléatoires

Nous pouvons donc afficher les questions et les réponses. Il nous reste à traiter le sujet bien plus complexe de la *sélection* de ces questions et réponses. Avant les commentaires détaillés, examinons le principe général de la méthode utilisée.

Nous voulons sélectionner une question et sa réponse correcte correspondante, puis remplir le tableau QU avec cinq nombres, représentant les positions dans le fichier principal des cinq réponses potentielles, y compris la réponse correcte. La question et la réponse principales sont d'abord choisies au hasard dans la totalité du fichier. Le numéro de la question dans le tableau principal est placé dans une position aléatoire du tableau QU.

Après quoi, il reste à trouver quatre autres réponses. Selon que nous voulons un test facile ou difficile, nous sélectionnons les quatre réponses soit à partir de la totalité du fichier principal, soit à partir de la section contenant des réponses de même type que la réponse principale. Les quatre réponses sont choisies au hasard dans la section appropriée du fichier, avec deux vérifications : que la même réponse ne figure pas deux fois et qu'il n'y a pas de réponse paraissant identique à la réponse correcte.

MODULE 4.4.14 : LIGNES 18000-18210

```
18000 REM ********************
18010 REM *Sélection Aléatoire*
18020 REM ********************
18030 mainq=INT(RND*(it))
18040 ctype=VAL(LEFT$(tableau$(mainq,0),
1))
18050 r1=ntype(1,ctype)
18060 r2=ntype(0,ctype)
18070 IF r2<5 OR rq=2 THEN r1=0:r2=it
18080 qpos=INT(RND*5)
18090 FOR i=0 TO 4
18100 duff=mainq
18110 dup=1:WHILE dup=1 AND i<>qpos
18120 dup=0
18130 duff=r1+INT(RND*r2)
18140 IF MID$(tableau$(duff,0),2)=MID$(t
ableau$(mainq,0),2) THEN dup=1
18150 FOR j=0 TO i-1
18160 IF MID$(tableau$(duff,0),2)=MID$(t
ableau$(qu(j),0),2) THEN dup=1
18170 NEXT j
18180 WEND
18190 qu(i)=duff
18200 NEXT i
18210 RETURN
```

Commentaires

Lignes 18030-18040 : La question et la réponse principales sont sélectionnées et leur position est stockée dans MAINQ. Le type de réponse est enregistré par la variable CTYPE.

Lignes 18050-18070 : Les deux variables R1 et R2 enregistrent le début et la fin de la partie du fichier d'où proviennent les questions. Si l'utilisateur a spécifié le type de test le plus difficile, R1 et R2 sont définies de façon à pointer sur le début et sur la fin du groupe de questions de même type que la question principale. S'il y a moins de cinq questions dans ce groupe, de sorte qu'il serait impossible de choisir cinq réponses différentes, ou si l'utilisateur a spécifié la forme de test la plus facile, R1 et R2 sont définies pour pointer sur le début du fichier. Si vous spécifiez la forme de test la plus difficile et si le programme ne la fournit pas, la raison probable est qu'il n'y a pas assez de réponses du même type que la question principale.

Ligne 18080 : QPOS est la position de la réponse correcte dans le tableau de cinq réponses possibles.

Lignes 18090-18200 : Cette boucle choisit les quatre autres réponses dans la partie du fichier indiquée par R1 et R2, chacune stockée temporairement dans la variable DUFF. Dans la boucle, des vérifications comparent la nouvelle réponse à la réponse correcte, qui peut se trouver n'importe où dans QU, et aux réponses placées précédemment dans QU. Notez qu'il ne suffit pas de contrôler que la même réponse du fichier principal n'est pas en double. Deux questions provenant de parties différentes du fichier principal peuvent fort bien avoir des réponses identiques. A la fin de la boucle, QU contient les positions des cinq réponses différentes dans le fichier principal.

Test

Le seul test efficace de ces modules exige une quantité suffisante de données pour permettre des tests significatifs. Comme premier test rapide, vous pourriez enregistrer deux types de questions : TYPE1, TYPE2, ... TYPE 6. Puis entrer une série de questions sous la forme Q1, Q2, ... Q10 avec des réponses A1, A2, ... A10. Le type des cinq premières questions devrait être TYPE1, les questions restantes ayant le TYPE2, ... TYPE6. Vous avez ainsi un jeu de questions capables de produire les formes les plus difficiles de test et cinq autres avec une seule question pour chacune.

Appelez le générateur de questions aléatoires et spécifiez la forme de test la plus difficile. Vous devez pouvoir continuer à répondre aux questions, tout en étant correctement informé de l'exactitude ou de l'inexactitude de vos réponses. Lorsqu'une question est choisie parmi les cinq premières, les cinq réponses doivent être comprises entre 1 et 5. Lorsque d'autres questions principales sont choisies, les réponses doivent provenir du fichier global de 10 questions.

Module 4.4.15 : Calcul du résultat

Pour finir, nous voulons que le programme puisse donner la note ou le résultat des tests. C'est moins facile qu'il n'y paraît, car il ne suffit pas de prendre le nombre de bonnes réponses et d'en tirer un pourcentage du total.

Si l'utilisateur spécifie simplement la première réponse de chaque test, ce sera en moyenne la bonne réponse pour une question sur cinq. Donc un résultat de 20 % peut fort bien indiquer que l'utilisateur n'a pas la moindre idée de la réponse correcte. La solution consiste à soustraire un cinquième des questions totales (le nombre que l'on peut attribuer à la chance pure) des réponses justes, et à exprimer ce nombre en pourcentage des quatre cinquièmes du nombre total de questions.

MODULE 4.4.15 : LIGNES 19000-19110

```
19000 REM **********
19010 REM *Résultat*
19020 REM **********
19030 CLS
19040 PRINT "Résultat:":PRINT
19050 IF total=0 THEN x$="********* PAS
ENCORE DE RESULTAT *********":GOSUB 2500
0:RETURN
19060 PRINT "Total des réponses:";total
19070 PRINT "Réponses correctes:";juste
19080 PRINT:PRINT "Résultat:";INT((juste
-total/5)/(total*0.8)*100+0.5);"%"
19090 PRINT:INPUT "Voulez-vous repartir
à zéro (o/n)";q$
19100 IF LOWER$(q$)="o" THEN total=0:jus
te=0
19110 RETURN
```

Test

Exécutez à nouveau le test du module précédent. Après avoir répondu à quelques questions, revenez au menu principal et appelez le module de résultat. Vous devriez constater que le résultat donné est assez logique, même s'il n'est pas facile de faire la relation exacte avec le nombre de bonnes réponses. Vous devez également pouvoir ramener le résultat à zéro et recommencer un nouveau test.

Si ce test donne satisfaction, le programme est opérationnel.

5

Vos Finances

Dans ce dernier chapitre, nous allons nous intéresser à un aspect important non encore abordé : le 464 et l'argent. Un sujet qu'il serait dommage d'ignorer car les micro-ordinateurs traitent remarquablement les problèmes financiers. Les sommes ainsi traitées sont rarement importantes, sinon ce ne serait pas l'affaire d'un micro peu coûteux, et les calculs sont généralement simples : additions et soustractions pour enregistrer les entrées et les sorties d'argent.

Mais, direz-vous, le cerveau humain peut fort bien faire cela. C'est vrai, mais le micro-ordinateur présente l'avantage de stocker l'information, de vous la fournir rapidement et de la présenter sous une forme immédiatement compréhensible. Les deux programmes de ce chapitre donnent un bon exemple de cette capacité; le premier vous permet de suivre votre compte en banque et le second de tenir une comptabilité simple.

Voici les deux programmes de ce chapitre :

BANQUIER : Permet d'enregistrer les mouvements d'argent de votre compte en banque.

COMPTABLE : Tient une comptabilité simple sous une forme classique.

PROGRAMME 5.1 : BANQUIER

Fonction du programme

Ce programme vous permet une mise à jour claire et continue de votre compte en banque, les références des mouvements, leur date et leur montant. Vous pouvez spécifier non seulement des opérations simples, mais des dépenses ou des recettes répétitives, avec ou sans périodicité régulière. La période considérée par le programme est une année normale.

```
                    JANVIER
-------------------------------------------
JOUR & DETAILS              SOMME     SOLDE
-------------------------------------------
   Solde                               0.00
 2 ALIMENTATION            456.70-   456.70-
 4 ESSENCE                 234.50-   691.20-
15 SALAIRE                5678.90   4987.70
17 TELEPHONE               511.10-  4476.60
19 EMPRUNT                2340.00-  2136.60
21 GARAGE                  123.40-  2013.20
26 ELECTRICITE             345.60-  1667.60
```

Figure 5.1 : Exemple d'écran de Banquier.

Module 5.1.1 : Initialisation

Il s'agit d'un module d'initialisation standard.

MODULE 5.1.1 : LIGNES 10000-10160

```
10000 REM ****************
10010 REM *Initialisation*
10020 REM ****************
10030 MODE 1
10040 WHILE in=0
10050 in=1:cr$=CHR$(13):
10060 DIM a$(99,1),a(99,1):a(0,1)=999
10070 RESTORE
10080 DIM mo$(11)
10090 FOR i=0 TO 11
10100 READ mo$(i)
```

```
10110 NEXT i
10120 DATA Janvier,Février,Mars,Avril,Ma
i,Juin
10130 DATA Juillet,Aout,Septembre,Octobr
e,Novembre,Décembre
10140 WEND
10150 INK 0,24:INK 1,3:INK 2,12
10160 WINDOW #1,1,40,16,25
```

Commentaires

Ligne 10060 : Le tableau A$ permet de stocker les noms des mouvements et une chaîne spéciale, que nous expliquerons plus tard, qui enregistre les mois où le mouvement particulier a eu lieu. Le tableau numérique A contient le montant de chaque mouvement et le jour du mois où il a lieu. Une définition de A(0,1) à 999, soit un quantième impossible pour le mois, permettra à la routine de tri ultérieure, de détecter la fin du fichier.

Lignes 10080-10130 : Cette boucle lit les noms et les mois de l'année dans le tableau MO$.

Module 5.1.2 : Le menu de programme

C'est un module de menu standard, qui de plus vous informe si le programme doit fournir des résultats avant même d'avoir entré des données.

MODULE 5.1.2 : LIGNES 11000-11210

```
11000 REM ******
11010 REM *Menu*
11020 REM ******
11030 CLS:CLS #1:PEN 2:PRINT TAB(16);"BA
NQUIER";TAB(16);"========":PEN 1:WINDOW
1,40,4,25
11040 z=0:WHILE z<>6:CLS
11050 PRINT "Commandes disponibles:":PRI
NT
11060 PRINT "1) Nouveaux mouvements"
11070 PRINT "2) Consulter/Supprimer les
mouvements"
11080 PRINT "3) Afficher le relevé"
```

```
11090 PRINT "4) Sauvegarder le fichier"
11100 PRINT "5) Charger le fichier"
11110 PRINT "6) Fin"
11120 PRINT:INPUT "Que voulez-vous:",z
11130 WHILE pa=0 AND (z=2 OR z=3 OR z=4)
11140 PRINT:PRINT:PRINT"   ***** PAS ENC
ORE DE DONNEES *****":SOUND 1,1000,100:F
OR i=1 TO 1500:NEXT i
11150 z=0
11160 WEND
11170 ON z GOSUB 12000,13000,14000,15000
,16000
11180 WEND
11190 CLS
11200 LOCATE 11,11:PRINT "FERME POUR AFF
AIRES"
11210 END
```

Module 5.1.3 : Entrée de nouveaux articles

Ce module d'entrée est plus complexe que les précédents, tout simplement parce qu'il s'agit d'entrées plus complexes. Pour chaque article enregistré, il faut connaître cinq éléments : si le mouvement est un crédit ou un débit (une recette ou un paiement), le nom du mouvement, le montant, les mois où ce mouvement doit avoir lieu et le jour du mois où il est effectué.

MODULE 5.1.3 : LIGNES 12000-12400

```
12000 REM **************************
12010 REM *Entrer nouveaux articles*
12020 REM **************************
12030 CLS
12040 PRINT "Nouveaux articles:":PRINT
12050 PRINT "1) Crédit":PRINT "2) Débit"
12060 PRINT:INPUT "Lequel voulez-vous:",
cd:cd=cd-1
12070 q$="":WHILE q$="" OR LEN(q$)>18:PR
INT:PRINT "Nom du mouvement (max. 18 car
s)":INPUT ":",q$:WEND
12080 PRINT:INPUT "Montant:",q
12090 en=1
12100 WHILE en
```

```
12110 PRINT:INPUT "Mois (e.g.01040710):"
,r$:PRINT
12120 en=0:FOR i=1 TO LEN(r$) STEP 2
12130 m=VAL(MID$(r$,i,2))-1
12140 IF m<0 OR m>11 THEN PRINT:PRINT "
   ***** MOIS ENTRE INCORRECT *****":SOUN
D 1,1000,100:en=1:i=LEN(r$):FOR j=1 TO 1
500:NEXT j
12150 IF en=0 THEN PRINT mo$(m);"/";
12160 NEXT i:PRINT:PRINT
12170 WEND
12180 INPUT "Jour du mouvement:",s
12190 PRINT:INPUT "Est-ce correct (o/n)"
;t$
12200 IF LOWER$(t$)<>"o" THEN RETURN
12210 pa=pa+1
12220 j=pa-1
12230 WHILE s<a(ABS(j),1) AND j>=0
12240 FOR k=0 TO 1
12250 a$(j+1,k)=a$(j,k)
12260 a(j+1,k)=a(j,k)
12270 NEXT k
12280 j=j-1
12290 WEND
12300 j=j+1
12310 a$(j,1)="000000000000"
12320 FOR i=1 TO LEN(r$) STEP 2
12330 m=VAL(MID$(r$,i,2))
12340 a$(j,1)=LEFT$(a$(j,1),m-1)+"1"+RIG
HT$(a$(j,1),12-m)
12350 NEXT i
12360 a$(j,0)=q$
12370 a(j,0)=q
12380 a(j,1)=s
12390 IF cd=1 THEN a(j,0)=-a(j,0)
12400 RETURN
```

Commentaires

Lignes 12040-12060 : Le programme doit savoir clairement si le mouvement est un paiement (débit) ou une recette (crédit). Ce point est enregistré dans la variable CD (Crédit/Débit).

Lignes 12090-12170 : Les mois où le mouvement doit avoir lieu sont entrés sous la forme d'une chaîne de nombres de deux chiffres, sans séparation. Donc, si un mouvement doit

être effectué en février, mai, août et novembre, vous auriez '02050811' représentant les mois 2, 5, 8 et 11. La boucle FOR commençant à la ligne 12120 vérifie que la chaîne ne comprend que des valeurs de mois réalistes et vous signale toute erreur éventuelle. A titre de vérification supplémentaire, la boucle affiche les noms des mois spécifiés tels qu'ils sont enregistrés dans MO$, afin que vous puissiez contrôler qu'il s'agit bien des mois prévus. La routine de la ligne 12090 se répète si un mois erroné a été entré, puisque la boucle WHILE dépend de la valeur de EN (Error Number + Numéro d'erreur) qui contient une valeur en cas d'entrée incorrecte. Notez la fenêtre utilisée pour l'entrée concernant les mois, qui permet d'effacer les lignes particulières concernées, sans perdre de vue ce qui a déjà été entré.

Ligne 12210 : A ce stade, l'information entrée a été vérifiée et confirmée par l'utilisateur, donc la variable qui enregistre le nombre de données dans le fichier, PA, est augmentée de 1.

Lignes 12230-12300 : Cette boucle permet de placer la nouvelle donnée dans le fichier, dans l'ordre de jour correct. Plutôt que de faire plusieurs copies des mouvements qui se répètent pour plus d'un mois, la méthode retenue consiste à stocker toutes les entrées une fois seulement, dans l'ordre des *jours*. Lorsqu'un relevé pour un mois particulier est demandé, un autre segment du programme explore toutes les entrées, à la recherche de celles qui coïncident avec le mois spécifié ou qui le précèdent, et qui sont donc à prendre en compte pour le calcul du solde.

A l'insertion de la nouvelle donnée, la boucle des lignes 12230 à 12290 commence avec la valeur de jour la plus haute (soit le nombre fictif 999 inséré dans le module d'initialisation) et redescend jusqu'à ce qu'elle trouve un mouvement avec une valeur de jour *inférieure* à la valeur de S, la valeur du jour de l'article que vous venez d'entrer. Si elle ne trouve *pas* la position correcte, elle déplace l'article qu'elle vient d'examiner d'une position vers le haut. Autrement dit, pendant l'exploration du fichier, elle déplace une ligne de réserve. Dès que la position correcte est trouvée, la ligne de réserve est déjà en bonne position. Pour finir, la valeur de J, qui enregistre la position de la première entrée trouvée avec un jour de mouvement inférieur à S, est augmentée de 1 pour pointer sur la ligne de réserve.

Lignes 12310-12350 : La tâche suivante consiste à traduire la liste des mois, entrée sous une forme facilement analysable par la suite du programme. La méthode retenue repose sur une

chaîne de 12 zéros, sur l'enregistrement des mois auxquels le mouvement *ne doit pas* être effectué, puis sur l'utilisation d'une boucle pour remplacer un zéro par un, pour tous les mois où le mouvement *doit* avoir lieu. Donc, dans le cas de notre exemple trimestriel précédent, nous obtiendrions la chaîne '010010010010'.

Lignes 12360-12390 : La nouvelle information est placée dans les tableaux principaux. Si la variable CD indique que le mouvement est un débit, le montant est multiplié par moins un pour le rendre négatif.

Test

Exécutez le programme et choisissez l'option 1 du menu.

Entrez un nouvel article de la façon suivante :

Débit (option 2)
Nom : TEST
Montant : 100
Mois : 02050811
Jour : 15

Après un certain délai, le menu principal réapparaît. Arrêtez le programme avec l'option 5. Puis entrez :

PRINT A$(0,0),A$(0,1),A(0,0),A(0,1)

Vous devez obtenir :

TEST 010010010010 -100 15

Module 5.1.4 : Affichage du relevé

Bien qu'il reste encore des modules à venir, la tâche finale de la partie principale du programme consiste à produire un relevé pour l'un des mois de l'année, à partir des articles entrés au module précédent. Le relevé comporte un solde antérieur des mois précédents et affiche en détail tous les mouvements du mois, ainsi que le nouveau solde après chaque mouvement.

MODULE 5.1.4 : LIGNES 14000-14290

```
14000 REM ********
14010 REM *Relevé*
14020 REM ********
14030 somme=0
14040 CLS:PRINT "Relevé:":PRINT
```

```
14050 q=0:WHILE q<1 OR q>12
14060 PRINT:INPUT "Numéro du mois du rel
evé (1-12):",q
14070 WEND
14080 FOR j=1 TO q-1
14090 FOR i=0 TO pa-1
14100 IF MID$(a$(i,1),j,1)="1" THEN somm
e=somme+a(i,0)
14110 NEXT i
14120 NEXT j
14130 PRINT:INPUT "Envoyer le relevé à l
'imprimante (o/n)";p$:ch=0:IF LOWER$(p$)
="o" THEN ch=8
14140 CLS:PRINT #ch,TAB (20-LEN(mo$(q-1)
)/2);UPPER$(mo$(q-1)):PRINT #ch,STRING$(
39,"-")
14150 PRINT #ch,"JOUR & DETAILS";TAB(25)
;"ARTICLE      SOLDE"
14160 PRINT #ch,STRING$(39,"-")
14170 PRINT #ch,"   Solde à nouveau";:PE
N 2-SGN(somme+0.001):PRINT #ch,TAB (32);
USING "####.##-";somme
14180 FOR i=0 TO pa-1
14190 d=0:WHILE MID$(a$(i,1),q,1)="1" AN
D d=0:d=1
14200 PEN 1:PRINT #ch,USING "## &";a(i,1
);a$(i,0);
14210 PEN 2-SGN(a(i,0+0.001)):PRINT #ch,
TAB (23);USING "####.##-";a(i,0);
14220 somme=somme+a(i,0)
14230 PEN 2-SGN(somme+0.001):PRINT #ch,T
AB (32);USING "####.##-";somme
14240 WEND
14250 NEXT i:PEN 1
14260 IF ch=8 THEN RETURN
14270 PRINT:PRINT "Frapper une touche po
ur continuer:"
14280 WHILE INKEY$="":WEND
14290 RETURN
```

Commentaires

Ligne 14030 : La variable SOMME contient le solde du compte : le solde antérieur et celui consécutif à chaque mouvement.

Lignes 14080-14120 : A condition qu'il ne s'agisse pas du

relevé du premier mois, pour lequel il n'y a pas de solde antérieur, ces deux boucles explorent la liste complète des mouvements, une fois pour chaque mois qui précède le mois du relevé. Ainsi, chaque mouvement est examiné, pour voir s'il existe à l'un des mois précédents, auquel cas le montant adéquat est ajouté au contenu de SOMME. Au terme des deux boucles, SOMME contient le total global de toutes les variations de solde depuis le début de l'année. (Pour avoir un solde complet, y compris les reliquats éventuels du début de l'année, il suffit d'entrer le solde à la fin de l'année précédente, sous forme d'un crédit, le premier janvier).

Ligne 14170 : A noter que si le contenu de SOMME est négatif, la couleur d'impression devient rouge. Vous retrouverez souvent cette même technique aux lignes suivantes. Notez également l'utilisation de la commande PRINT USING qui nous permet d'afficher le nombre sous un format standard, pour obtenir un relevé bien présenté avec les décimales correctement alignées.

Lignes 14180-14250 : Cette boucle explore la liste complète des mouvements, tandis que la boucle des lignes 14190-14240 ne sélectionne que ceux qui ont un « 1 » dans la position de la chaîne qui enregistre les mois où le mouvement est attendu. Si un mouvement doit avoir lieu pendant le mois spécifié pour le relevé, la boucle affiche le jour, A(I,1), le nom, A$(I,0), le montant, A(I,0), et finalement le solde résultant du mouvement, obtenu en ajoutant le montant au contenu de SOMME. A l'exception du jour, les opérations de débits apparaissent en rouge. Les colonnes de l'écran restent cohérentes, malgré la longueur variable des nombres, grâce à TAB, qui donne une position standard de départ sur l'écran, et à PRINT USING.

Test

Entrez les données de test utilisées au module précédent. Lorsque le menu principal réapparaît, choisissez l'option 3 pour afficher le relevé.

Commencez par le premier mois et, si vous êtes satisfait, revenez au menu principal et affichez le relevé du mois suivant. Bien sûr, la majorité des relevés seront vierges, puisque la plupart des mois n'ont pas de mouvement. Toutefois, les mois de février, mai, août et novembre ont tous un mouvement qui doit être affiché.

Module 5.1.5 : Fichiers de données

Il s'agit d'un module de fichier de données standard.

MODULE 5.1.5 : LIGNES 15000-16140

```
15000 REM *************************
15010 REM *Sauvegarde sur cassette*
15020 REM *************************
15030 CLS:q$="":WHILE LOWER$(q$)<>"o"
15040 INPUT "Nom du fichier à sauvegarde
r:",fi$
15050 PRINT:PRINT "Fichier à sauvegarder
 ";fi$
15060 PRINT:INPUT "Est-ce correct (o/n)"
;q$
15070 WEND
15080 OPENOUT fi$
15090 PRINT #9,pa
15100 FOR i=0 TO pa-1
15110 PRINT #9,a$(i,0);cr$;a$(i,1);cr$;a
(i,0);cr$;a(i,1)
15120 NEXT i
15130 CLOSEOUT
15140 RETURN
16000 REM *******************
16010 REM *Lire sur cassette*
16020 REM *******************
16030 CLS:q$="":WHILE LOWER$(q$)<>"o"
16040 INPUT "Nom du fichier à lire";fi$
16050 PRINT:PRINT "fichier à lire ";fi$
16060 PRINT:INPUT "Est-ce correct (o/n)"
;q$
16070 WEND
16080 OPENIN fi$
16090 INPUT #9,pa
16100 FOR i=0 TO pa-1
16110 INPUT #9,a$(i,0),a$(i,1),a(i,0),a(
i,1)
16120 NEXT i
16130 CLOSEIN
16140 RETURN
```

Module 5.1.6 : Modification et suppression de données

Comme dans Nnombre, c'est un module de modifications par l'utilisateur, basé cette fois-ci sur une boucle FOR qui explore les articles un à un.

MODULE 5.1.6 : LIGNES 13000-13250

```
13000 REM *********************
13010 REM *Consulter/Supprimer*
13015 REM *   les articles    *
13020 REM *********************
13030 FOR i=0 TO pa-1
13040 CLS
13050 PRINT "Mouvement:";a$(i,0)
13060 PRINT "Montant:";a(i,0)
13070 PRINT "Mois:";
13080 FOR j=1 TO 12
13090 IF MID$(a$(i,1),j,1)="1" THEN PRIN
T mo$(j-1);"/";
13100 NEXT j:PRINT
13110 PRINT "Jour de mouvement:";a(i,1)
13120 PRINT:PRINT "Commandes disponibles
:":PRINT
13130 PEN 2:PRINT "ENTER";:PEN 1:PRINT "
 article suivant"
13140 PRINT "'ç' quitter"
13150 PRINT "'çD' Supprimer l'article"
13160 q$="":PRINT:INPUT "Lequel voulez-v
ous:",q$
13170 WHILE UPPER$(q$)="çD"
13180 FOR j=i TO pa-1
13190 FOR k=0 TO 1
13200 a$(j,k)=a$(j+1,k):a(j,k)=a(j+1,k)
13210 NEXT k,j
13220 pa=pa-1:q$="ç"
13230 WEND
13240 IF q$="" THEN NEXT i
13250 RETURN
```

Test

Après avoir entré et sauvegardé sur cassette certaines données, appelez cette option. Vous devez pouvoir consulter les articles et les supprimer. Si ces fonctions sont possibles, le programme est opérationnel.

PROGRAMME 5.2 : COMPTABLE

Fonction du programme

Le deuxième programme de ce dernier chapitre est plus complexe que Banquier. Il a pour fonction de tenir les deux parties d'un jeu de comptes simples, présentés sous la forme classique, avec certains articles autonomes et d'autres clairement divisés en groupes représentant différents types de dépenses. Dans son état actuel, le programme peut traiter des sommes allant jusqu'à F 9999.99. Au-delà, les calculs sont corrects mais le format des comptes n'est pas bon. S'il vous faut des montants ou des totaux égaux ou supérieurs à 10 000, vous devrez apporter quelques modifications simples aux commandes PRINT USING et à l'espacement de l'affichage.

Module 5.2.1 : Initialisation

Il s'agit d'un module d'initialisation standard.

MODULE 5.2.1 : LIGNES 10000-10080

```
10000 REM ****************
10010 REM *Initialisation*
10020 REM ****************
10030 CLS
10040 WHILE in=0
10050 in=1:cr$=CHR$(13):
10060 DIM a$(1,99),a(1,99)
10070 WEND
10080 MODE 1
```

```
                    DEBIT

FRAIS DE VOITURE
   REPARATION              98.21
   ESSENCE                198.56
   PNEUS                   54.00
                          -------
                                        350.77
AMSTRAD CPC 464                         349.00

MAISON
   GAZ                     45.87
   ELECTRICITE             56.43
   DIVERS                 126.75
                          -------
```

```
                                    229.05
VACANCES                            364.76
                                   -------
TOTAL:                             1293.58
```

Figure 5.2 : Exemple d'écran de Comptable.

Commentaires

Ligne 10060 : Les deux parties des comptes, crédit et débit, avec les noms associés à chaque opération, sont stockées des deux côtés des tableaux A et A$. Vous pouvez stocker jusqu'à 100 articles des deux côtés, et même facilement augmenter ce nombre.

Module 5.2.2 : Le menu principal

Il s'agit d'un module de menu standard qui interdit certaines fonctions avant que des données n'aient été entrées.

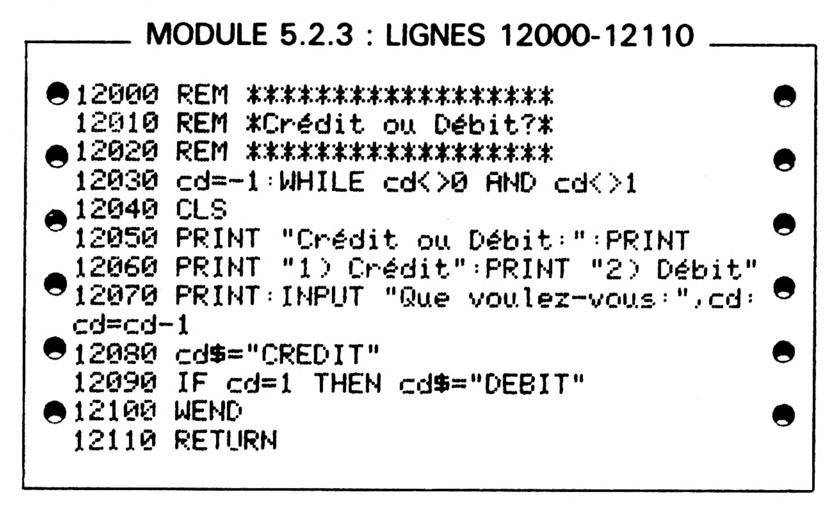
MODULE 5.2.2 : LIGNES 11000-11220

```
11000 REM ******
11010 REM *Menu*
11020 REM ******
11030 CLS:PEN 2:PRINT TAB(15);"COMPTABLE
";TAB(15);"=========":PEN 1:WINDOW 1,40,
4,25
11040 z=0:WHILE z<>6:CLS
11050 PRINT "Commandes disponibles:":PRI
NT
11060 PRINT "1) Nouveaux postes"
11070 PRINT "2) Changer/Supprimer l'arti
cle"
11080 PRINT "3) Afficher les comptes"
11090 PRINT "4) Sauvegarder le fichier"
11100 PRINT "5) Charger le fichier"
11110 PRINT "6) Fin"
11120 PRINT:INPUT "Que voulez-vous:",z
11130 IF z<4 AND z>0 THEN GOSUB 12000
11140 WHILE c(cd)=0 AND (z=2 OR z=3 OR z
=4)
11150 PRINT:PRINT:PRINT"***** PAS ENCORE
 DE DONNEES *****":SOUND 1,1000,100:FOR
i=1 TO 1500:NEXT i
11160 z=0
```

```
11170 WEND
11180 ON z GOSUB 13000,16000,18000,19000
,20000
11190 WEND
11200 CLS
11210 LOCATE 11,11:PRINT "PROGRAMME TERM
INE"
11220 END
```

Module 5.2.3 : Crédit ou débit?

Contrairement à Banquier, plusieurs parties de ce programme doivent savoir s'il s'agit d'un crédit ou d'un débit, aussi la routine qui demande cette information se trouve dans un module séparé.



MODULE 5.2.3 : LIGNES 12000-12110

```
12000 REM ******************
12010 REM *Crédit ou Débit?*
12020 REM ******************
12030 cd=-1:WHILE cd<>0 AND cd<>1
12040 CLS
12050 PRINT "Crédit ou Débit:":PRINT
12060 PRINT "1) Crédit":PRINT "2) Débit"
12070 PRINT:INPUT "Que voulez-vous:",cd:
cd=cd-1
12080 cd$="CREDIT"
12090 IF cd=1 THEN cd$="DEBIT"
12100 WEND
12110 RETURN
```

Module 5.2.4 : Le type d'article

Bien que simpliste, ce module explique pourquoi ce programme est plus long qu'un programme du type Banquier. Ce module vous permet de spécifier l'un des trois types auxquels appartient un article à entrer. Voici les trois types :

1) Un article simple : Le nom et le montant de l'article suffisent. Lors de l'impression du compte, les articles individuels apparaîtront à gauche et les montants dans la colonne principale de nombres à droite.

2) Un poste principal : C'est ce type qui permet de spécifier des groupes d'articles dans le compte global. Par exemple, si vous utilisiez le programme pour tenir une comptabilité domestique, vous pourriez choisir 'VOITURE' comme poste principal d'un groupe d'articles comprenant les pneus, l'essence, les réparations, etc... Sur le compte, le nom du poste principal apparaîtra à gauche, mais il n'y aura pas de montant en face lui.

3) Sous-postes : Comme nous venons de l'indiquer, chaque poste principal peut être suivi d'une liste d'articles qui constituent un groupe séparé. Dans les comptes, les noms des sous-postes apparaîtront sous leur poste principal correspondant, décalés vers la droite, et les montants associés à chaque sous-poste apparaîtront à gauche de la colonne principale de nombres.

MODULE 5.2.4 : LIGNES 13000-13120

```
13000 REM ******************
13010 REM *Entrée des Postes*
13020 REM ******************
13030 CLS
13040 PRINT "Nouveaux articles:":PRINT
13050 PRINT cd$:PRINT
13060 PRINT "L'artile est-il:":PRINT
13070 PRINT "1) Un simple article;"
13080 PRINT "2) Un poste principal ou"
13090 PRINT "3) Un sous-poste"
13100 PRINT:INPUT "(0 pour quitter)";typ
e
13110 ON type GOSUB 14000,14000,15000
13120 RETURN
```

Test

Après avoir entré toutes les parties du programme dans lesquelles n'intervient aucun calcul, il vaut mieux exécuter le programme et tester rapidement le menu. Si vous spécifiez vouloir entrer un nouvel article, le programme doit vous demander si c'est un crédit ou un débit, puis le type; ce sera tout pour le moment. La seule autre fonction du menu active à ce stade, est l'option 5 permettant d'arrêter le programme. Le menu lui-même vous interdira l'accès aux fonctions de modification des données ou d'affichage des comptes, puisqu'aucune donnée n'a encore été entrée.

Module 5.2.5 : Entrée de simples articles et de postes principaux

Deux modules séparés traitent de l'entrée des sous-postes d'une part et des simples articles ou des postes principaux d'autre part. Pour bien comprendre la suite du programme, observez bien la façon dont les articles sont stockés et les caractères indicateurs spéciaux qui enregistrent le type de l'article.

MODULE 5.2.5 : LIGNES 14000-14120

```
14000 REM ******************
14010 REM *Simple article ou*
14015 REM * Poste Principal *
14020 REM ******************
14030 q=0:q$="":r$=""
14040 PRINT:INPUT "Nom de l'article:",q$
14050 IF type=1 THEN PRINT:INPUT "Montan
t de l'article:",q
14060 PRINT:INPUT "Est-ce correct (o/n)"
;r$
14070 IF LOWER$(r$)<>"o" THEN PRINT:PRIN
T "     ******** NON ENREGISTRE ********"
:SOUND 1,1000,100:FOR i=1 TO 1500:NEXT i
:RETURN
14080 IF type=1 THEN q$="%"+q$ ELSE q$="
*"+q$
14090 a$(cd,c(cd))=q$
14100 a(cd,c(cd))=q
14110 c(cd)=c(cd)+1
14120 RETURN
```

Commentaires

Ligne 14050 : Comme nous l'avons vu dans l'introduction des modules précédents, les postes principaux ne sont pas accompagnés d'un montant, donc cette ligne n'accepte un montant que pour de simples articles.

Ligne 14080 : Il n'y a pas de zone de stockage distincte pour les différents types d'articles, mis à part les côtés crédit et débit des tableaux. La suite du programme déterminera le type d'article en consultant un caractère indicateur spécial précédant le début du nom de l'article. Ce sera '%' pour un simple article et '*' pour un poste principal.

Lignes 14090-14120 : Vous connaissez déjà la variable CD qui enregistre la nature crédit ou débit d'un article. Ici, CD per-

met de décider le côté des tableaux A et A$ où le nouvel article doit être placé. En outre, nous devons garder un enregistrement du nombre d'articles des côtés crédit et débit, puisqu'ils seront normalement différents. Cela est accompli par le tableau C. Le tableau n'a pas été déclaré dans le module d'initialisation car il n'aura que deux éléments, C0 et C1, correspondant aux côtés crédit et débit des tableaux principaux. A nouveau, la valeur de CD permet d'indiquer lequel des deux éléments de C doit être utilisé. A partir de là, nous pouvons voir que la référence suivante :

```
A    (CD,    C(CD))
1      2        3
```

signifie :

1) Un élément dans le tableau numérique A.
2) Du côté indiqué par la valeur de CD, c'est-à-dire crédit ou débit.
3) Le premier élément vide de ce côté, déterminé par ce qui est déjà stocké.

Test

Exécutez le programme et choisissez l'option des nouvelles entrées. Indiquez que vous voulez entrer un poste principal du côté crédit, puis entrez TEST PRINCIPAL comme nom d'article; aucune valeur ne doit être demandée. Sélectionnez à nouveau l'option des nouvelles entrées et spécifiez un simple article du côté crédit, avec le nom TEST et la valeur 100. Faites exactement la même chose du côté débit des comptes.

Arrêtez le programme à partir du menu et entrez, en mode direct :

PRINT A(0,10),A(1,0),A$(0,0),A$(1,0)[ENTER]

Vous devez voir :

0   0   *EST PRINCIPAL   *EST PRINCIPAL

Puis faites de même pour la ligne 1 des tableaux , par exemple A(0,1), etc. Vous devez obtenir :

100   100   %TEST   %TEST

Pour terminer, affichez la valeur de C(0) et C(1). Tous deux doivent être égaux à 2.

Module 5.2.6 : Entrée d'un sous-poste

L'entrée d'un nouveau sous-poste n'est pas aussi simple que celle d'un simple article. Pour chaque nouveau sous-poste

entré, il faut vérifier la présence du poste principal correspondant et placer l'article près de son poste principal au lieu de l'ajouter à la fin des articles déjà stockés.

MODULE 5.2.6 : LIGNES 15000-15210

```
15000 REM ************
15010 REM *Sous Poste*
15020 REM ************
15030 PRINT:INPUT "Poste principal:",q$
15040 q$="*"+q$
15050 pl=-1:FOR i=0 TO c(cd)-1
15060 IF a$(cd,i)=q$ THEN pl=i+1
15070 NEXT i
15080 IF pl=-1 THEN PRINT:PRINT "     ***
* PAS DE POSTE DE CE NOM ****":SOUND 1,1
000,100:FOR i=1 TO 1500:NEXT i:RETURN
15090 PRINT:INPUT "Nom du sous-poste:",q
$
15100 PRINT:INPUT "Montant:",q
15110 PRINT:INPUT "Est-ce correct (o/n)"
;r$
15120 IF LOWER$(r$)<>"o" THEN PRINT:PRIN
T "     ******** NON ENREGISTRE ********"
:SOUND 1000,100:FOR i=1 TO 1500:NEXT i:R
ETURN
15130 q$="$"+q$
15140 FOR i=c(cd)+1 TO pl+1 STEP -1
15150 a$(cd,i)=a$(cd,i-1)
15160 a(cd,i)=a(cd,i-1)
15170 NEXT i
15180 a$(cd,pl)=q$
15190 a(cd,pl)=q
15200 c(cd)=c(cd)+1
15210 RETURN
```

Commentaires

Lignes 15030-15050 : Le nom du poste principal concerné est demandé et il y a une vérification du contenu du fichier, pour voir si le poste existe ; si ce n'est pas le cas, un message d'erreur apparaît et le programme revient au menu.

Lignes 15140-15200 : Comme nous l'avons dit, un sous-poste doit apparaître dans les comptes principaux comme élément d'un groupe apparaissant sous le poste principal correspondant. La méthode simple pour cela, consiste à le stocker dans

le fichier près de son poste principal. La position du premier article suivant le poste principal a déjà été trouvée par la boucle FOR à la ligne 15050; il ne reste donc qu'à déplacer vers le haut tous les articles à partir de ce point et à placer le nouvel article au bon endroit du tableau.

Test

Entrez les articles spécifiés pour le test du module 5.2.5, puis rappelez le nouveau module d'entrée pour placer un nouveau sous-poste du côté crédit, appelé TEST SOUS, avec une valeur de 200. Faites de même pour le côté débit.

Entrez ce qui suit en mode direct :

FOR I=0 TO 2 :PRINT A(0,I),A(1,I),A$(0,I),A$(1,I) :NEXT I[ENTER]

Vous devez obtenir :

```
0      0      *EST PRINCIPAL      *EST PRINCIPAL
200    200    $TEST SOUS          $TEST SOUS
100    100    %TEST               %TEST
```

Affichez les valeurs de C(0) et de C(1), qui toutes deux doivent être égales à 3.

Module 5.2.7 : Fichiers de données

Comme les données du logiciel Comptable sont plutôt complexes, il vaut mieux entrer le module de fichier de données à ce stade, pour éviter de réentrer les données pendant le test. Après l'entrée du module, entrez et sauvegardez les données spécifiées pour le test du module précédent.

MODULE 5.2.7 : LIGNES 19000-20160

```
19000 REM *************************
19010 REM *Sauvegarde sur cassette*
19020 REM *************************
19030 CLS:q$="":WHILE LOWER$(q$)<>"o"
19040 INPUT "Nom du fichier à sauvegarde
r:";fi$
19050 PRINT:PRINT "Fichier à sauvegarder
 ";fi$
19060 PRINT:INPUT "Est-ce correct (o/n)"
;q$
19070 WEND
19080 OPENOUT fi$
19090 FOR i=0 TO 1
19100 PRINT #9,c(i)
```

```
19110 FOR j=0 TO c(i)-1
19120 PRINT #9,a$(i,j);cr$;a(i,j)
19130 NEXT j
19140 NEXT i
19150 CLOSEOUT
19160 RETURN
20000 REM **********************
20010 REM *Lecture sur cassette*
20020 REM **********************
20030 CLS:q$="":WHILE LOWER$(q$)<>"o"
20040 INPUT "Nom du fichier à charger:";
fi$
20050 PRINT:PRINT "Fichier à charger ";f
i$
20060 PRINT:INPUT "Est-ce correct (y/n)"
;q$
20070 WEND
20080 OPENIN fi$
20090 FOR i=0 TO 1
20100 INPUT #9,c(i)
20110 FOR j=0 TO c(i)-1
20120 INPUT #9,a$(i,j),a(i,j)
20130 NEXT j
20140 NEXT i
20150 CLOSEIN
20160 RETURN
```

Module 5.2.8 : Modifications des articles

Il s'agit d'un simple module avec quelques fonctions supplémentaires pour tenir compte du fait que certains articles ne sont pas isolés, mais font partie de groupes d'articles sous un poste principal commun.

MODULE 5.2.8 : LIGNES 16000-16290

```
16000 REM *****************************
16010 REM *Changements et suppressions*
16020 REM *****************************
16030 FOR i=0 TO c(cd)-1
16040 d=0:WHILE d=0:d=1
16050 CLS
16060 PRINT "Changer ou supprimer:":PRIN
T
16070 IF LEFT$(a$(cd,i),1)<>"$" THEN PRI
```

```
NT MID$(a$(cd,i),2);
16080 IF LEFT$(a$(cd,i),1)="*" THEN hh$=
MID$(a$(cd,i),2):PRINT
16090 IF LEFT$(a$(cd,i),1)="$" THEN PRIN
T hh$:PRINT "  ";MID$(a$(cd,i),2);
16100 IF a(cd,i)<>0 THEN PRINT TAB(32);U
SING"####.##";a(cd,i)
16110 PRINT:PRINT "Commandes disponibles
:":PRINT
16120 PRINT "1) Article suivant"
16130 PRINT "2) Changer le montant"
16140 PRINT "3) Retour au menu"
16150 PRINT "4) Supprimer l'article"
16160 PRINT:PRINT "Que voulez-vous?"
16170 q$="":WHILE q$=""
16180 q$=INKEY$
16190 WEND
16200 IF q$="4" THEN GOSUB 17000:RETURN
16210 IF q$="3" THEN RETURN
16220 WHILE q$="2" AND LEFT$(a$(cd,i),1)
<>"*"
16230 PRINT:INPUT "Montant à ajouter:",q

16240 PRINT:INPUT "Est-ce correct (o/n)"
;r$
16250 IF LOWER$(r$)="o" THEN a(cd,i)=a(c
d,i)+q:q$=""
16260 WEND
16270 WEND
16280 NEXT i
16290 RETURN
```

Commentaires

Lignes 16070-16090 : Si l'article lu dans le fichier est un simple article, il est affiché, bien que débarassé du caractère indicateur accolé au début du nom. S'il s'agit d'un poste principal, il est non seulement affiché, mais son nom est stocké dans HH$ afin qu'il puisse s'afficher au-dessus de tous les sous-postes qui suivent.

Lignes 16220-16260 : Outre la suppression d'articles, vous pouvez modifier la valeur associée à un poste. Pour cela, il faut entrer un nombre positif ou négatif qui doit modifier la valeur d'un article; non pas la nouvelle valeur absolue de l'article. L'avantage est que la plupart des modifications impliqueront d'ajouter des montants aux articles existants lorsque des

dépenses ou des recettes supplémentaires sont faites sous des articles déjà existants. Ainsi, si 1000 F de plus doivent être dépensés pour la réparation de la voiture, pour laquelle il y a déjà un poste, il suffit d'explorer le fichier jusqu'à ce poste et d'entrer '1000'.

Test

Exécutez le programme et lisez les données précédemment stockées sur cassette. Puis sélectionnez l'option 2 du menu et vérifiez que vous pouvez consulter trois articles, puis revenez au menu. Choisissez encore l'option 2 et cette fois-ci, essayez d'ajouter ou de soustraire des montants aux deux totaux précédemment entrés. Consultez à nouveau les articles pour constater que leurs valeurs ont bien été modifiées.

Module 5.2.9 : Suppression d'articles

Il nous reste à pouvoir supprimer des articles existants. Dans le cas présent, le module de suppression est plus complexe que les autres modules du même type. La raison en est l'existance de groupes formés avec des postes principaux. S'il n'y a pas de difficulté particulière pour supprimer un simple article ou un sous-poste, il n'en est pas de même pour un poste principal. Bien entendu, il faut non seulement supprimer le poste principal mais également tous les sous-postes qui lui sont associés. Sinon le compte présentera des sous-postes non associés à un poste principal, d'où un compte incohérent.

MODULE 5.2.9 : LIGNES 17000-17140

```
17000 REM ************
17010 REM *Suppression*
17020 REM ************
17030 pl=i:gr=1
17040 d=0:WHILE LEFT$(a$(cd,pl),1)="*" A
ND d=0:d=1
17050 WHILE LEFT$(a$(cd,pl+gr),1)="$"
17060 gr=gr+1
17070 WEND
17080 WEND
17090 FOR k=pl TO c(cd)-gr-1
17100 a(cd,k)=a(cd,k+gr)
17110 a$(cd,k)=a$(cd,k+gr)
17120 NEXT k
17130 c(cd)=c(cd)-gr
17140 RETURN
```

Commentaires

Ligne 17030 : La position où la suppression doit avoir lieu est reçue du module précédent grâce à la variable I. Elle est transférée dans PL pour les besoins du module en cours. La variable GR (GRoupe) indique combien d'articles sont à supprimer. Elle reçoit initialement la valeur 1 et ne sera augmentée que si l'article à supprimer est un poste principal avec des sous- postes associés.

Lignes 17040-17080 : Ces deux boucles imbriquées ne seront activées que si l'article à supprimer est un poste principal. La boucle interne analyse les entrées suivantes, comptant le nombre d'articles précédés d'un $, indiquant qu'il s'agit d'articles subordonnés au poste principal. Le résultat de ce calcul se trouve dans GR.

Lignes 17090-17120 : Une boucle classique de resserrement d'un tableau et de suppression d'un article. Toutefois, il y a ici une différence : au lieu de copier l'article X dans l'espace X-1 et, par conséquent, de copier chaque élément une position plus bas, les articles sont déplacés de GR positions, ce qui supprime GR articles, soit le nombre d'articles dans le groupe correspondant à un poste principal.

Test

En suivant la procédure de test du module précédent, vous pourrez non seulement consulter les articles et les modifier, mais également les supprimer. Si vous supprimez l'article appelé TEST PRINCIPAL, vous constaterez que TEST SOUS disparaît également.

Module 5.2.10 : Affichage des comptes

Après tout ce travail de préparation, ce module permet d'afficher le compte sous sa forme définitive. Comme le module équivalent de Banquier, il paraît complexe, mais dès que vous aurez vu l'affichage, tout s'éclairera.

MODULE 5.2.10 : LIGNES 18000-18310

```
18000 REM ************************
18010 REM *Affichage des comptes*
18020 REM ************************
18030 tt=0:ss=0
18040 CLS:PRINT "Afficher les comptes:":
PRINT
```

```
18050 PRINT:INPUT "Envoyer comptes à l'i
mprimante (o/n)";p$:ch=0:IF LOWER$(p$)="
o" THEN ch=8
18060 CLS
18070 PRINT #ch,TAB(20-LEN(cd$)/2);cd$:P
RINT
18080 FOR i=0 TO c(cd)-1
18090 tt=tt+a(cd,i)
18100 IF LEFT$(a$(cd,i),1)="*" THEN PRIN
T #ch
18110 IF LEFT$(a$(cd,i),1)="$" THEN PRIN
T #ch,"  ";
18120 PRINT #ch,MID$(a$(cd,i),2);
18130 d=0:WHILE LEFT$(a$(cd,i),1)<>"*" A
ND d=0:d=1
18140 PRINT #ch,TAB(18);
18150 IF LEFT$(a$(cd,i),1)="%" THEN PRIN
T #ch,TAB(29);
18160 PRINT #ch,USING "####.##";a(cd,i);
18170 IF LEFT$(a$(cd,i),1)="$" THEN ss=s
s+a(cd,i)
18180 WEND
18190 PRINT #ch
18200 WHILE ss<>0 AND LEFT$(a$(cd,i+1),1
)<>"$"
18210 PRINT #ch,TAB(18);"-------"
18220 PRINT #ch,TAB(29);USING "####.##";
ss
18230 ss=0
18240 WEND
18250 NEXT i
18260 PRINT #ch,TAB(29);"-------"
18270 PRINT #CH,"TOTAL:";TAB(29);USING "
####.##";tt
18280 IF ch=8 THEN RETURN
18290 PRINT:PRINT "Frapper sur une touch
e pour continuer"
18300 WHILE INKEY$="":WEND
18310 RETURN
```

Commentaires

Ligne 18090 : La variable TT permet de stocker le total cumulé du compte parallèlement à l'affichage.

Lignes 18100-18120 : Ces lignes affichent le nom de l'article. Pour un poste principal, une ligne vierge est affichée d'abord pour le séparer de ce qui précède, tandis que pour un sous-poste, le nom est décalé de deux espaces.

Lignes 18130-18180 : Ces lignes affichent les montants des simples articles et des sous-postes. Les sous-postes s'afficheront sur la 18ème position de la ligne et les simples articles sur la position 29. Si l'article traité est un sous-poste, le sous-total des articles du groupe en cours est stocké temporairement dans SS.

Lignes 18190-18230 : Cette boucle n'est active que lors du traitement d'un groupe ; en effet, si SS n'est pas égal à 0 et si l'article suivant ne fait pas partie du groupe, le groupe est complet. La boucle a pour effet d'afficher le total du groupe dans la colonne de nombres principale sur la position 29.

Test

Relisez les données précédemment stockées sur cassette et choisissez simplement l'option 3 du menu principal. Si ce test est satisfaisant, le programme est opérationnel.

CONTENU DETAILLE



IMPRIMERIE AUBIN, 86240 LIGUGÉ
Dépôt légal n° 2096-1-1986 — Imprimeur n° L 20974
Collection n° 50 — Édition n° 01

10-0059-5

MICROPRATIQUES

Activités avec l'Amstrad CPC 464

Hachette *Informatique*

Voici un ouvrage de programmes pour votre Amstrad, qui présente une triple possibilité d'utilisation :

- comme manuel pour l'apprentissage, par la pratique, de techniques avancées de programmation en BASIC, grâce aux commentaires ligne à ligne qui accompagnent les programmes bien structurés ;
- comme une collection de programmes, de qualité professionnelle, permettant de mettre à profit le BASIC puissant de l'Amstrad, et aux nombreuses applications pratiques : traitement de fichiers, finances et comptabilité, graphisme, enseignement assisté par ordinateur, jeux... Bien qu'écrits pour le CPC 464, les possesseurs de CPC 664 n'auront aucune difficulté à adapter ces programmes pour leur machine ;
- comme une importante bibliothèque de modules et sous-programmes réutilisables aisément dans un nombre illimité de programmes que vous désirez écrire.



F 125.00/86/1

10/0059/5

Imprimé en France
SUD OFFSET - 94 RUNGIS

Design Contours.
Illustration : Agnès Lévy.


Activités avec l'Amstrad CPC 464
Hachette


AMSTRAD
CPC
OCR
MÉMOIRE ÉCRITE
MEMORY ENGRAVED
MEMORIA ESCRITA
AC
ME
https://acpc.me/